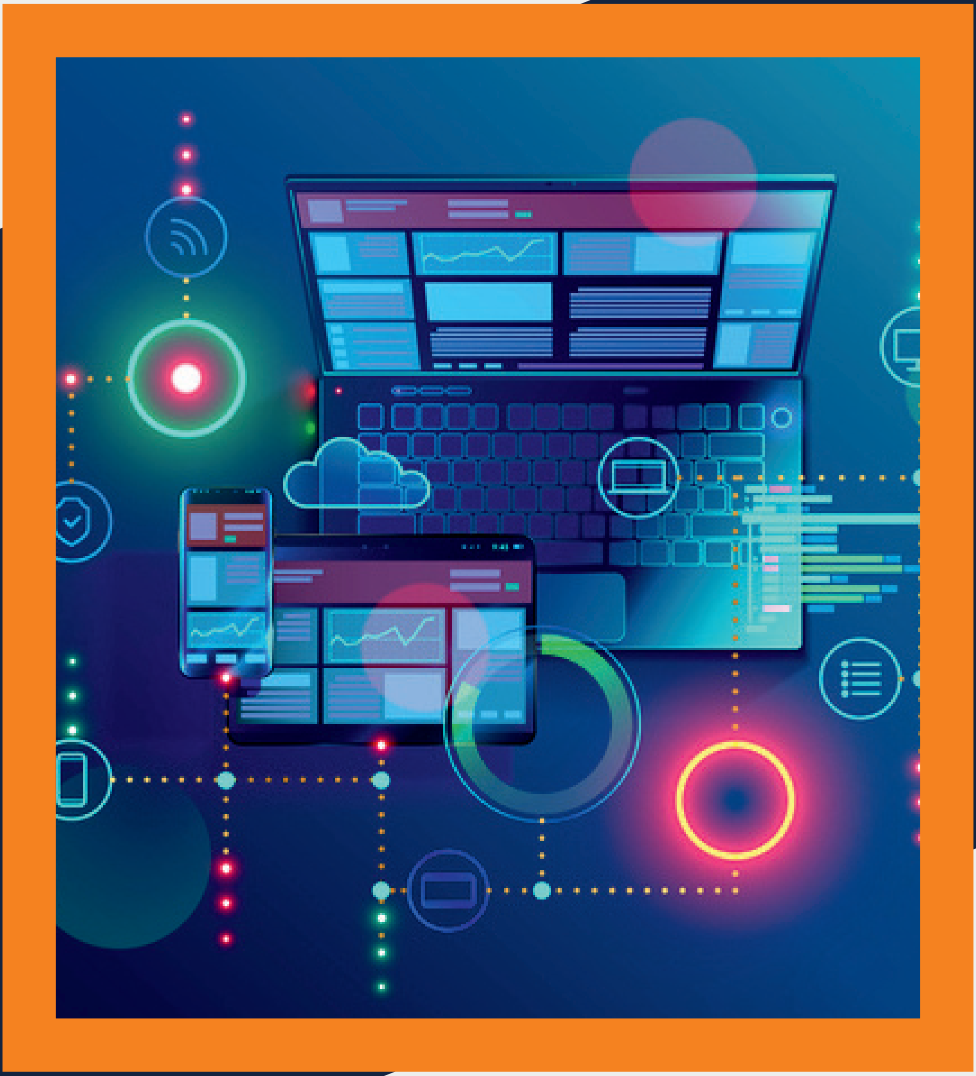
FUNDAMENTALS OF OPERATING SYSTEM



Rama Krishna K.
Dr. Vikas Sharma

ALEXIS PRESS
JERSEY CITY, USA

FUNDAMENTALS OF OPERATING SYSTEM

FUNDAMENTALS OF OPERATING SYSTEM

Rama Krishna K.

Dr. Vikas Sharma

ALEXIS PRESS

ALEXIS PRESS

Published by: Alexis Press, LLC, Jersey City, USA
www.alexispress.us

First Published 2022

A catalogue record for this publication is available from the British Library

Library of Congress Cataloguing in Publication Data

Includes bibliographical references and index.

Fundamentals of Operating System by *Rama Krishna K., Dr. Vikas Sharma*

ISBN 978-1-64532-866-7

CONTENTS

CHAPTER 1
AN OVERVIEW OF OPERATING SYSTEM

Mr. Rama Krishna K., Assistant Professor
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- ramakrishna@presidencyuniversity.in

A computer device is made up of several components that aid in its operation and processing. There are five main components of a computer that contribute to make data processing simpler and more comfortable. The fundamental computer components and their operations. Sample questions based on this idea have also been provided for individuals practicing Computer Knowledge for forthcoming competitive examinations. Components of a computer system are the essential pieces that enable an electronic device to operate smoothly and quickly. There are many fundamental components, which include: An operating system is made up of numerous components that work together to complete certain tasks. Though most operating systems vary in form, they share logical components. Each component must be a well-defined part of a system that explains the functions, inputs, and outputs adequately[1].

An operating system consists of the eight components listed below:

A. Process Management
B. I/O Device Management
C. File Management
D. Network Management
E. Main Memory Management
F. Secondary Storage Management
G. Security Management
H. Command Interpreter System

The next section goes through all of the aforementioned components in further detail:

Process Management

A process is a program or a portion of a program stored in main memory. To do its purpose, a process requires resources like as CPU time, memory, files, and I/O devices. The process management component handles the various processes that are executing on the operating system at the same time.A running programme is referred to as a process.

In terms of process management, the operating system is in charge of the following tasks:

Processes may be created, loaded, executed, suspended, resumed, and terminated.

Switch the system between numerous processes running in main memory.

Provides communication methods that allow processes to communicate with one another.

Provides techniques for controlling concurrent access to shared data in order to keep shared data consistent.

Properly allocate and de-allocate resources to avoid or avert a stalemate scenario.

I/O Device Management

One of the functions of an operating system is to conceal from the user the idiosyncrasies of individual hardware components. I/O Device Management abstracts H/W devices while retaining application data to assure correct device utilisation, eliminate mistakes, and give users with a comfortable and efficient programming environment.

The following are the responsibilities of the I/O Device Management component:

Hide the specifics of H/W devices.

Manage device main memory using cache, buffers, and spooling.

Keep track of and supply custom drivers for each device.

File Management

One of the most apparent functions provided by an operating system is file management. Computers may store data in a variety of physical formats, the most prevalent of which being magnetic tape, disc, and drum. A file is defined by the author of the file as a collection of associated information. Files are often used to represent data, source and object forms, and programmes. Data files might be alphabetic, numeric, or alphanumeric in nature. A file is a collection of bits, bytes, lines, or records whose meaning is determined by the author and user. The operating system puts the abstract idea of the file into action by handling mass storage devices like types and drives. In addition, files are often grouped into folders to facilitate their usage. These folders may contain files as well as other directories[2].

In terms of file management, the operating system is in charge of the following tasks:

A. File and directory creation and deletion
B. Support for primitives for manipulating files and directories
C. File mapping to secondary storage
D. File backup on nonvolatile (stable) storage medium

Network Management

Because network administration encompasses multiple diverse components, the definition of network management is often wide. The process of managing and administering a computer network is known as network management. A computer network is a collection of several kinds of computers that are linked to one another. Network management includes fault investigation, maintaining service quality, network provisioning, and performance management. Network administration is the act of maintaining your network healthy so that multiple machines can communicate efficiently.

The following are network management characteristics:

A. Network management
B. Network maintenance
C. Network operation
D. Network provisioning

Main Memory Management

Memory is a big collection of words or bytes, each with its own unique address. It is a cache of frequently accessed data shared by the CPU and I/O devices.

Main memory is a volatile storage medium, which means that its data are lost in the event of a system failure or when the system power is turned off.

Memory Management's primary goal is to maximise memory usage on the computer system.

The operating system is responsible for the following memory management activities:

A. Keeping track of which regions of memory are presently being utilised and by whom.
B. Select which processes should be loaded when memory space becomes available.
C. As required, allocate and deallocate memory space[3].

Secondary Storage Management

A computer system's primary function is to run programmes. These programmes, as well as the data they access, must be executed in main memory. Because main memory is insufficient to store all data and programmes permanently, the computer system must offer secondary storage to back up main memory.

Disks are the primary on-line storage media for both programmes and data in the majority of current computer systems. Most programmes, such as compilers, assemblers, sort routines, editors, and formatters, are saved on disc until loaded into memory, and then utilise the disc as both the source and destination of their processing.

In terms of disc management, the operating system is in charge of the following tasks:

A. Management of available space
B. Storage planning

Security Management

The operating system is essentially in charge of all tasks and activities that occur in a computer system. The many processes in an operating system must be kept separate from one another. Various approaches may be used to guarantee that only programmes that have received valid authorization from the operating system can access files, memory segments, CPU, and other resources. Security Management is a system for restricting the access of programmes, processes, or people to resources specified by computer restrictions to be enforced, as well as some means of enforcement. Memory addressing hardware, for example, ensures that a process may only run inside its own address area. The timer ensures that no process gains CPU control without surrendering it. Finally, no process is permitted to do its own I/O in order to safeguard the integrity of the different peripheral devices.

System of Command Interpretation

The command interpreter is a critical component of every operating system. The principal interface between the user and the rest of the system is the command interpreter. The Command Interpreter System carries out a user command by invoking one or more underlying system applications or system calls. The Command Interpreter System enables human users to interface with the operating system while also providing a comfortable programming environment. Control statements provide several commands to the operating system. A programme that reads and interprets control statements is run automatically. This software is known as the shell, and some examples include the Windows DOS command window, the Bash of Unix/Linux, and the C-Shell of Unix/Linux[3].

Other Significant Activities

An operating system is a complicated piece of software. Apart from the aforementioned components and duties, the Operating System performs a variety of additional tasks. Here are a few examples:

A. Security it prevents unwanted access to programmes and data by using passwords and other similar measures.
B. System performance control by recording the time between a service request and a system response.
C. Job accounting Keeping track of the time and resources used by different tasks and users.
D. Error-detection tools Create dumps, tracing, error messages, and other debug and error-detection tools.

Coordination of other software's and users Coordination and assignment of compilers, interpreters, assemblers, and other software to computer system users.

Execution of Instructions:

Memory is made up of a huge array of words or bytes, each with its unique address. The CPU retrieves instructions from memory based on the programme counter value. These instructions may result in extra memory addresses being loaded and stored. For example, in a normal instruction-execution cycle, an instruction is initially read from memory. The instruction is subsequently decoded, which may result in the retrieval of operands from memory.

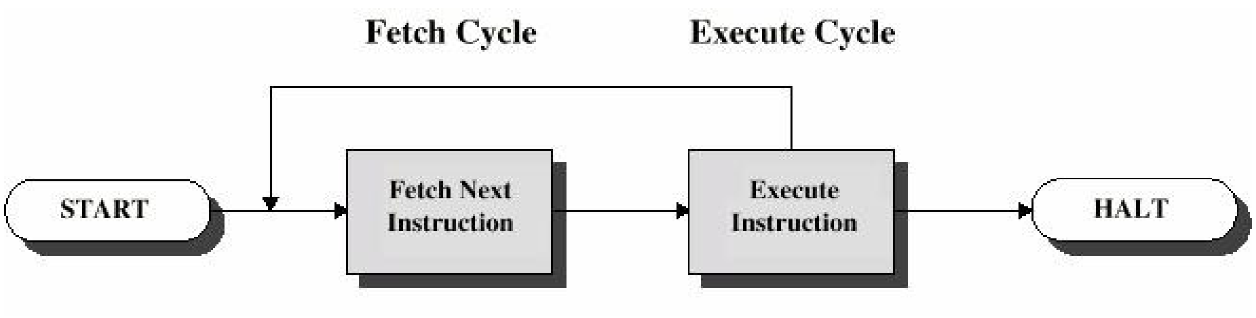


Figure: 1.1: Flow diagram of instructions execution.

After executing the instruction on the operands, the results may be stored in memory. The memory unit sees a stream of memory addresses but has no idea how they are formed (by the instruction counter, indexing, indirection, literal addresses, etc.) or what they are for (instructions or data). As a result, we may disregard a program that creates a memory address. We are only interested in the memory address sequence created by the executing application[4].

The six stages necessary to execute a single instruction are summarized here.

Step 1: Retrieve the instruction.

Step 2: Decode the instruction and get the operands.

Step 3: Carry out the ALU function.

Step 4: Read from memory.

Step 5: Save the result to the register file.

Step 6: Update the computer.

Interrupt

An interrupt is a signal sent by a device connected to a computer or by a software running on the computer. It is necessary for the operating system (OS) to pause and determine what to do next. An interrupt is a temporary halt or termination of a service or a current operation. For this reason, most I/O devices provide a bus control line called Interrupt Service Routine (ISR). An interrupt signal may be scheduled (i.e., explicitly requested by a programme) or unplanned (i.e., produced by an event unrelated to a programme presently executing on the system).Almost all computer systems today are interrupt-driven. This implies that they follow a set of computer instructions in a programme and execute the instructions until they reach the end or detect an interrupt signal. If the latter occurs, the computer either continues the current programme or starts another one. In either instance, activities must be halted until the next course of action is determined. The operating system use pauses in operations to do this and operate on other applications.When the device processor handles interrupts, it notifies the device that sent the signal that the interrupt request (IRQ) was recognized. An interrupt handler is normally included in an operating system to prioritise interrupts and store them in a queue if more than one is waiting to be handled. It also features a scheduler software that selects which programme will take control next.If an interruption occurs, the connected service may not start right away. Interrupt latency is the time delay between when an interrupt occurs and when ISR execution begins[5], [6].

Interrupts of many kinds

There are two kinds of interruptions:

Hardware interrupt

A hardware interrupt is an electrical signal sent by an external hardware device informing the operating system that it requires attention. Moving a mouse or pushing a keyboard key is one example of this. In these interrupt scenarios, the CPU must pause to receive the mouse location or input at that time. All devices are linked to the Interrupt Request Line in this form of interrupt (IRL). A hardware IRQ often has a value that links it with a specific device. By increasing the IRQ, the CPU is able to recognise which device is seeking service and then give it appropriately.

Hardware interruptions are classified into three types:

Interrupts that may be disguised

An internal interrupt mask register in a CPU selectively allows and disables hardware requests. The interrupt is enabled when the mask bit is set. When it is clear, the interrupt is turned off. Maskable interrupts are signals that are impacted by the mask.

Non-maskable interrupts

Because the interrupt mask cannot be deactivated in certain instances, it does not effect some interrupt signals. These are interruptions that are not maskable and are generally high-priority events that cannot be ignored[7].

Spurious interrupts

A spurious interrupt, also known as a phantom interrupt or ghost interrupt, is a sort of hardware interrupt for which no source can be discovered. If a system malfunctions, these interruptions are difficult to detect. If the ISR does not account for the likelihood of such interruptions, the system may get stuck.

Software interrupts

When an application program stops or requires specific services from the operating system, a software interrupt occurs. When particular circumstances are satisfied by executing a special instruction, the CPU typically requests a software interrupt. This instruction causes the interrupt and acts similarly to a subroutine call. When the system interacts with device drivers or when an application seeks OS services, software interrupts are widely employed. Software interruptions may be generated unexpectedly by program execution faults rather than by design in certain situations. These interruptions are referred to as exceptions or traps.

Interrupts vs. polling

Polling is a continuous monitoring condition in which a microcontroller in a computer system examines the status of all devices on a regular basis. The first device found with the IRQ bit is serviced first, and the appropriate ISR is called to do so. The system is simple to set up and guarantees that the component that need service receives it.

However, since the microcontroller spends all of its processing time polling, it cannot do many operations at the same time. In addition, probing the IRQ bit of all devices wastes a lot of time. This issue is solved with interrupts. The controller does not need to routinely check the state of devices while using interrupts. Instead, it only replies when an interruption occurs. As a result, when an interrupt occurs, the controller is told that it requires servicing[8].

ISR

For each interrupt, an ISR is present. Each ISR's address is kept at a fixed position in memory. An interrupt service routine (ISR) may request asynchronous interrupts and handle both mask able and non-mask able interrupts.

When an interrupt occurs, the ISR is executed by the microcontroller. At the start of execution, the ISR disables all other devices' interrupt services, pausing the presently processed instruction and preserving its configuration in a register. The interrupt's programme counter will then be loaded from a position specified by the interrupt vector table. When the ISR is finished, it will reinitialize all interrupt services. There are two kinds of interrupt handlers:

A. Interrupt handler at the first level (FLIH). A hard or fast interrupt handler with jitter throughout process execution that handles mask able interrupts.
B. Interrupt handler at the second level (SLIH). FLIH is a slow and gentle interrupt handler with reduced jitter.

Interrupt method types

Interrupt triggering mechanisms or modules are classified into two types:

Interrupts induced by a change in level

By holding the interrupt signal at a certain active logic level, this interrupt module creates an interrupt. After the device has been serviced, the signal is negated by the CPU. The interrupt signal is sampled by the CPU throughout each instruction cycle and recognised when it is entered during the sampling process. Before departing the ISR, the processor may serve additional devices after serving one. Using wired-OR connections, level-triggered interruptions enable numerous devices to share a same signal. However, dealing with them might be difficult for firmware.

Edge-triggered interrupts

This module creates an interrupt only when it detects the interrupt source's asserting edge. In this case, the device wishing to indicate an interrupt generates a pulse that alters the interrupt source level. Edge-triggered interrupts give a more flexible method of handling interrupts. Irrespective of how the interrupt source acts, they may be served instantly. Furthermore, they lessen the complexities of firmware code and the number of conditions needed for the firmware. The disadvantage is that specific hardware may be needed to identify and service the interruption if the pulse is too brief to be detected.

Memory organization

The computer memory hierarchy is shaped like a pyramid and is used to explain the distinctions between memory types. It divides computer storage into hierarchical levels.

Level 0: CPU registers

Level 1: Cache memory

Level 2: Main memory or primary memory

Level 3: Magnetic disks or secondary memory

Level 4: Optical disks or magnetic types or tertiary Memory

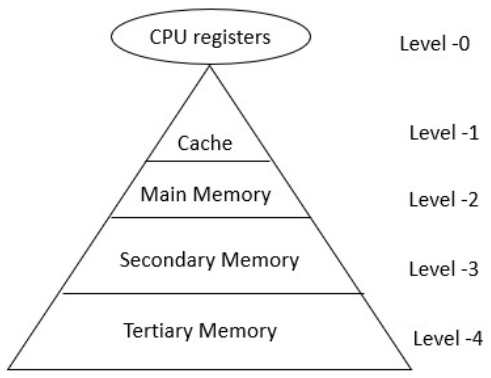


Figure 1.2: Memory Organization

Memory capacity and cost are inversely related to speed in Memory Hierarchy. The devices are organised here in the order Fast to Slow, that is, from register to Tertiary memory.

Registers at Level 0

The registers are found inside the CPU. They have the shortest access time since they are located inside the CPU. Registers are the most costly and have the lowest size, usually measured in kilobytes. Flip-Flops are used to implement them[9].

Cache at Level 1

Cache memory is used to store software parts that the CPU often accesses. It is more costly and smaller in size, often measured in Megabytes, and is accomplished utilising static RAM.

Primary or Main Memory at Level 2

It connects with the CPU and auxiliary memory devices directly through an I/O processor. Main memory is less costly than cache memory and has a bigger capacity, measured in gigabytes. Dynamic RAM is used to implement this memory.

Secondary storage at Level 3

Level 3 contains secondary storage devices such as magnetic discs. They are used for backup purposes. They are less expensive than main memory and greater in size, often in the TB range.

Tertiary storage at Level 4

At level 4, tertiary storage systems such as magnetic tape are available. They are the cheapest and biggest in size and are used to store removable data (1-20 TB).

Memory importance:

Memory hierarchy is the arrangement of several types of storage on a computer system depending on access speed. The most performance storage is CPU registers, which are the quickest to read and write to. Following cache memory is standard DRAM memory, followed by disc storage with varying degrees of performance such as SSD, optical, and magnetic disc drives. Hardware designers are increasingly depending on memory at the top of the memory hierarchy to close / minimize the performance gap in order to bridge the processor memory performance gap.

This is accomplished by expanding the size of cache hierarchy,minimizing reliance on slower main memory. In systems, memory hierarchy is employed. Memory hierarchy is the arrangement of several types of storage on a computer system depending on access speed. The most performance storage is CPU registers, which are the quickest to read and write to. Following cache memory is standard DRAM memory, followed by disc storage with varying degrees of performance such as SSD, optical, and magnetic disc drives. Hardware designers are increasingly depending on memory at the top of the memory hierarchy to close / minimise the performance gap in order to bridge the processor memory performance gap. This is accomplished by expanding the size of cache hierarchy (which processors can reach considerably quicker), minimising reliance on slower main memory.

Computer Cache Memory

Cache memory is memory that caches data from the CPU. Your computer stores data that the CPU need access to. For example, if you regularly search for a certain term, your computer will retain that word close to the material you are looking for. For example, if you open and then close Microsoft Excel. After a while, you opened it again. It will now take less time to open since it was pulled up from cache memory.

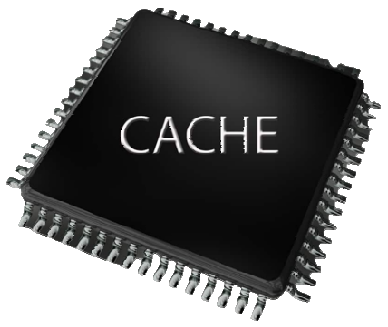


Figure 1.3: Computer Cache Memory

CPU Cache Function

To maximise performance, the CPU must have access to ultra-fast memory. This memory is used by the CPU cache, and when the CPU needs to retrieve data, it first searches in the cache memory. If the data is not discovered in cache memory, it is referred to as a miss; if it is found, it is referred to as a hit. Then it proceeds to search the main memory. If no data is discovered, it is retrieved from main memory and a copy of that data is placed into cache memory. So that the next time the same data is requested, it will be accessible in cache memory. If the data is not found in the main memory, it is sought in the secondary memory. This secondary memory search will use the majority of the time. The data is read from RAM and sent to the CPU cache by the memory controller. The memory hierarchy occurs in the CPU cache as well[9].

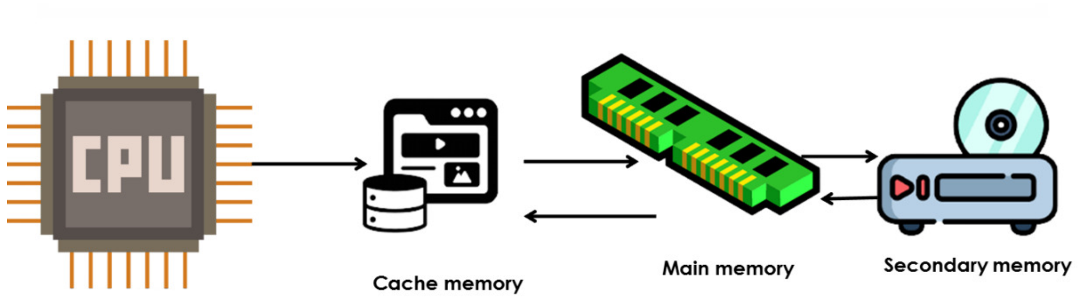


Figure 1.4: CPU Cache Function

Cache memory types in computers

L1 cache (or first-level cache): The first cache level, L1, stores data that the CPU has recently utilised. It caches data from recently utilised CPU cores for rapid access by other portions of the system. Because it is near to the CPU, data in this region is regularly accessible. L1 caches, which can generally carry 4 megabytes of data, are present on the majority of CPUs today.

Cache at Level 2 (L2): The L2 cache stores data that the CPU has recently utilised and will most likely use again. It is mostly found on graphics processing units (GPUs) that utilise vector instructions, such as those seen in games and 3D imaging applications.

The cache at level 3 (L3): It is the third cache level.The L3 memory cache was originally located on the motherboard. It was a long time ago, and most CPUs were single-core. The CPU's L3 cache may now be massive, with top-end consumer CPUs boasting up to 32MB of L3 cache. Some server CPU L3 caches can go over and above this limit by up to 64MB. The L3 cache is the biggest cache storage unit, but it is also the slowest.

Disc cache: Cache memory may also be found on your hard disc. The "disc cache" is what this is termed. Because the data is loaded from disc to memory, it is the slowest cache level. RAM may also store information about peripherals like computer devices and DVD drives in the cache of the peripheral.

GPU Cache: The data required to display the image must be obtained rapidly, hence a caching mechanism makes logical. When computer graphics are integrated, they are processed on a single chip by a graphics processing unit (GPU) and a CPU. Both functions use the same resource, which limits the GPU cache.

Cache memory has an impact on computer performance

Cache memory is a sort of memory that is used to store data that the computer often uses. This data is regularly retrieved and stored at a place near to the processor. The cache memory speeds up the computer by storing data and delivering it to the CPU when required. This lowers the need to look for the data on the storage device. The data in the cache memory is updated as often as feasible. The cache memory may assist minimise the amount of time spent on activities by boosting the computer's speed. This may assist to increase the computer system's overall performance. So, if you're looking to purchase a laptop or computer, you'll want more RAM/main memory since more RAM will boost the performance of your running processes or programmes. As more apps are run on the computer.

Cache memory mapping

Because your computer stores information in several places, it need an address to tell the components where to look for the data. Cache memory mapping is the process of mapping a segment of a computer's system memory to a physical memory location. In other words, it keeps frequently accessed material on the computer's hard drive in a more accessible position. This may boost system performance by allowing data to be accessed more rapidly by the computer's operating system and application applications. The system BIOS or software running on the computer performs cache memory mapping. When the BIOS or software creates the cache memory mapping, it instructs the operating system and application applications to begin utilising the mapped area of system memory rather than the resident memory. Cache memory mapping is useful in a variety of circumstances where frequently requested data has to be stored in a more accessible format. When the computer is running many programmes, each one requires access to various areas of the system memory. Cache memory mapping may be used to increase system speed by making data more rapidly available to the operating system and application applications on the computer.

Direct Memory Access (DMA)

Direct Memory Access (DMA) is a feature of various computer bus designs that enables data to be transmitted directly from a connected device (such as a disc drive) to memory on the computer's motherboard. Typically, a certain region of memory is designated as an area to be utilised for direct memory access, freeing the CPU from involvement in data transmission and thereby speeding up overall computer function. Up to 16 megabytes of memory may be accessed for DMA using the ISA bus standard. The EISA and Micro Channel Architecture standards provide access to the whole range of memory addresses (provided they can be addressed with 32 bits). DMA is accomplished via peripheral component connectivity by utilising a bus master (the CPU "delegating" I/O control to the PCI controller).

• The Programmed Input/Output (PIO) interface is an alternative to DMA in which all data sent between devices passes via the CPU. Ultra DMA is a newer protocol for the ATA/IDE interface that allows a burst data transfer rate of up to 33 MB (megabytes) per second. Ultra DMA/33 hard drives additionally support PIO modes 1, 3, and 4, as well as multiword DMA mode 2. (at 16.6 megabytes per second).

DMA Transfer Types

Memory to Memory Transfer: In this mode, a data block from one memory location is relocated to another. In this mode, the current address register of channel 0 is used to point the source address, and the current address register of channel is used to point the destination address. In the first transfer cycle, a data byte from the source address is loaded into the DMA controller's temporary register, and the data from the temporary register is stored in the memory pointed by the destination address in the next transfer cycle. Following each data transmission, the current address registers are decremented or increased in accordance with the current parameters. After each data transmission, the channel 1 current word count register is decremented by one. When the word count on channel 1 reaches FFFFH, a TC is issued, which triggers the EOP output and terminates the DMA service.

Initialize automatically

During startup in this mode, the microprocessor loads the base address and word count registers with the current address and word count registers. Throughout the DMA service, the address and count in the base registers stay constant. Once the first block transfer, i.e. after the EOP signal is activated, the original values of the current address and current word count registers are automatically restored from that channel's base address and base word count register. Without CPU interaction, the channel is ready to conduct another DMA service after auto startup.

DMA Processor

All DMA data transfers are managed by the controller, which is incorporated into the processor board. Data must be transferred between system memory and a 110 device in two phases. Data is sent from the transmitting device to the DMA controller, which subsequently sends it to the receiving device. The microprocessor informs the DMA controller of the data's location, destination, and quantity. The data is then sent via the DMA controller, enabling the CPU to proceed with other processing duties. When a device wants to transmit or receive data over the Micro Channel bus, it competes with all other devices attempting to obtain control of the bus. Arbitration is the name given to this procedure. Instead of the DMA controller arbitrating for BUS control, the I/O device delivering or receiving data (the DMA slave) participates in arbitration. When the central arbitration control point approves the DMA slave's request, the DMA controller assumes control of the bus.

Multicore: The processor/CPU is the computer's brain, and it is in charge of all operations. The core is the execution unit of the CPU. The basic purpose of the core is to read and execute instructions. These instructions may take several forms, such as computation, data transfer, branching, and so on.A Unicore CPU is a processor with a single core. A multicore processor, on the other hand, has two or more cores. A multicore processor's cores may read and execute programme instructions independently at the same time. This accelerates programme execution and allows for parallel processing. Quadcore CPUs, Octacore processors, and so on.Multicore processors are utilised in networking, embedded systems, digital signal processing (DSP), and graphics applications (GPU).

Benefits of Multicore

Multicore processors provide the following advantages:

A. Multicore processors can handle more work than unicore CPUs.
B. Multicore processing cores are housed on a single integrated chip. As a consequence, the system's clock speed rises.
C. When compared to a unicore CPU, multicore processors can tolerate more errors.
D. Although the applications are assigned distinct cores, they may nevertheless communicate with one another.
E. Multicore processors give better performance while using less energy, making them energy efficient.

Drawbacks of Multicore

The downsides of multicore processors are as follows:

A. Despite the presence of numerous cores in a multicore processor, the speed does not rise much as compared to simple processors.

B. Increasing the number of cores will increase the amount of software interferences caused by resource sharing.
C. When multitasking, the CPU generates tremendous heat; multicore processors use more electricity.

Multiprocessor

Two or more processors are used in multiprocessor systems. A multiprocessor system's processors share memory, a system bus (a data and information-carrying channel that links the key components of a computer system), and I/O devices. Multiprocessors may carry out many instructions at the same time. The failure of one processor has no effect on the operation of the others. As a consequence, multiprocessors have increased reliability.

Shared memory or distributed memory may be used in multiprocessor systems. Each CPU in a shared memory multiprocessor shares main memory and I/O devices. Using the same system bus, the CPUs may access the main memory. Symmetric multiprocessors are multiprocessors that employ shared memory.

In a distributed memory multiprocessor, each processor has its own private memory where computing takes place. The system bus enables processors to interact with one another and to access main memory.

The Benefits of a Multiprocessor

The following are the benefits of a multiprocessor system:

A. Because numerous processors are operating at the same time, the system's throughput rises.
B. Multiprocessor systems are more dependable since one processor's failure does not influence the others.
C. Parallel processing is accomplished because many processes are conducted concurrently on distinct processors.

Advantages and disadvantages of multiprocessor

The drawbacks of a multiprocessor system are as follows:

A. The coordination between a multiprocessor system's processors is highly difficult.
B. A common I/O device is used by all CPUs. So, if a process uses I/O, other processes must wait for their turn. As a consequence, the system's throughput may suffer.
C. Because all processors share the same memory, a large pool of main memory is necessary for efficient calculation.
D. Multiprocessor systems are more costly than multicore computers.

An operating system serves as a communication interface (bridge) between the user and computer hardware. An operating system's objective is to offer a platform for a user to run applications in a convenient and efficient way. An operating system is a piece of software that regulates computer hardware allocation. Hardware coordination must be suitable to guarantee the proper operation of the computer system and to prevent user applications from interfering with the normal operation of the system. As an example, much as a manager sends commands to his employees, we request or convey our orders to the Operating System. The primary purpose of the operating system is to make the computer environment more user-friendly, and the secondary goal is to utilize resources as efficiently as possible.

REFERENCES

[1] M. O. Farooq and T. Kunz, “Operating systems for wireless sensor networks: A survey,” Sensors, 2011, doi: 10.3390/s110605900.

[2] T. Glatard et al., “Reproducibility of neuroimaging analyses across operating systems,” Front. Neuroinform., 2015, doi: 10.3389/fninf.2015.00012.

[3] V. DiLuoffo, W. R. Michalson, and B. Sunar, “Robot Operating System 2,” Int. J. Adv. Robot. Syst., 2018, doi: 10.1177/1729881418770011.

[4] H. Studiawan, F. Sohel, and C. Payne, “A survey on forensic investigation of operating system logs,” Digital Investigation. 2019. doi: 10.1016/j.diin.2019.02.005.

[5] P. K. Hitigala Kaluarachchilage, C. Attanayake, S. Rajasooriya, and C. P. Tsokos, “An analytical approach to assess and compare the vulnerability risk of operating systems,” Int. J. Comput. Netw. Inf. Secur., 2020, doi: 10.5815/ijcnis.2020.02.01.

[6] K. Vdovjak, J. Balen, and K. Nenadić, “Experimental evaluation of desktop operating systems networking performance,” Int. J. Electr. Comput. Eng. Syst., 2020, doi: 10.32985/IJECES.11.2.2.

[7] N. Tsolakis, D. Bechtsis, and D. Bochtis, “Agros: A robot operating system based emulation tool for agricultural robotics,” Agronomy, 2019, doi: 10.3390/agronomy9070403.

[8] I. Ungurean, “Timing comparison of the real-time operating systems for small microcontrollers,” Symmetry (Basel)., 2020, doi: 10.3390/SYM12040592.

[9] O. Hahm, E. Baccelli, H. Petersen, and N. Tsiftes, “Operating Systems for Low-End Devices in the Internet of Things: A Survey,” IEEE Internet Things J., 2016, doi: 10.1109/JIOT.2015.2505901.

[10] F. R. A. Hopgood and A. J. T. Colin, “Introduction to Operating Systems,” Math. Gaz., 1973, doi: 10.2307/3615198.

[11] Benchpartner, “History, Introduction and Generation of Operating System,” 2020.

CHAPTER 2

EVOLUTION OF OPERATING SYSTEM

Mr. Rama Krishna K., Assistant Professor
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- ramakrishna@presidencyuniversity.in

An operating system (OS) is a piece of software that controls all other application program in a computer after being originally loaded by a boot program. Through a specified application program interface, the application program use the operating system by requesting services (API). Additionally, using a user interface, such as a command-line interface (CLI) or a graphical user interface (UI), users can communicate with the operating system directly (GUI). An operating system (OS) serves as a conduit between computer hardware and users. An operating system is a piece of software that manages files, memory, processes, input, output, and peripheral devices like disc drives and printers. It also handles input and output. An operating system is a piece of software that permits software program to communicate with a computer's hardware. The kernel is the name of the piece of software that houses the fundamental elements of the operating system [1]. A few of the most well-known operating systems are OS/400, Linux, AIX, Windows, VMS, z/OS, etc. Operating systems are widely used today like Toys mobile phones, automobiles, personal computers, mainframe computers, TV etc. The domain of system software includes operating systems. In essence, it controls the computer's resources. An operating system serves as an interface between the software and various computer hardware or software components. The operating system is made to be able to control all of the computer's resources and activities. It is an entirely integrated collection of specialized applications that manages all of the computer's functions. All other program that are installed on the computer, including application programs and other system software, are controlled and monitored by it. Windows, Linux, Mac OS, and other operating systems serve as examples[2].

Implement of an operating system:

The development of software for computers benefits greatly from an operating system. Without an operating system, each program would have to contain both its own user interface (UI) and the complete code required to manage all low-level computer operations, such as disc storage, network connections, and other things. This would greatly increase the size of any application and render software development difficult given the wide variety of underlying hardware available. Instead, a lot of routine operations, such transmitting a network packet or showing text on a display or other conventional output device, can be delegated to system software, which acts as a bridge between applications and hardware. Applications can interface with the hardware in a consistent and predictable manner without having to be aware of any specifics about the hardware thanks to the system software. That system software the operating system can support nearly any number of apps as long as they all access the same services and resources in the same way. By using a standard and understandable interface, users may control, configure, and maintain the system hardware, greatly reducing the time and coding needed to design and debug an application.

Objectives of the operating systems:

A. Usability is used for the improving the computer system's efficiency while increasing usability is one of the goals.
B. Making a computer system more user-friendly means adding an easier-to-use interface and increasing interactivity.
C. To function as a bridge between the hardware and its users in order to give consumers simple access to resources.
D. For controlling a computer's resources.
E. Controls and monitoring are used by keeping track of which users are utilizing which resources, approving resource requests, and resolving disputes arising from requests made by various users and program.
F. Ensuring effective and equitable resource sharing between consumers and program[3].

Characteristics of the operating systems:

A. Device management, often known as the Input/output controller, chooses which process receives the device, when it does, and for how long.
B. Resource allocation and de-allocation are handled via file management, together with selection of recipients.
C. The tracking of resources and time consumed by various jobs or users is known as job accounting.
D. Error-detecting the creation of dumps, traces, error warnings, and various debugging and error-detecting techniques are among the methods found in Aids.
E. Memory management allocates memory when a process or program demands it. It also keeps a record of the primary memory, such as which portions are used by whom and which portions are not.
F. When a task is finished or when it is no longer necessary, Processor Management de-allocates the processor it had been assigned to.
G. The system's response time to service requests is tracked by Control on System Performance.

Passwords or other forms of protection are used in security to prevent unauthorized access toprograms and data[4].

An operating system is a software that runs application applications and serves as a communication link (interface) between the user and computer hardware. The primary function of an operating system is to allocate resources and services, such as memory, devices, processors, and information. A traffic controller, a scheduler, a memory management module, I/O programmes, and a file system are among the programmes included in the operating system to manage these resources. Important features of an operating system include:

Security: To secure user data, the operating system employs password protection and other similar measures. It also protects applications and user data from illegal access.

System performance control: Monitors overall system health to assist enhance performance. To get a thorough picture of the system's health, record the time between service requests and system responses. This may assist to increase performance by giving critical information required to solve issues.

Accounting for jobs: This information may be used to measure resource utilization for a specific user or group of users by keeping track of the time and resources utilised by different tasks and users.

Error detecting aids: The operating system continually monitors the system to identify errors and prevent a computer system from malfunctioning.

Coordination of other software and users: Operating systems coordinate and allocate interpreters, compilers, assemblers, and other software to computer system users.

Memory Management: The operating system is in charge of the Primary Memory, also known as Main Memory. Main memory is composed of a vast array of bytes or words, with each byte or word allocated a unique address. Main memory is rapid storage that the CPU may access directly. A programme must first be loaded into main memory before it can be run. For memory management, an operating system performs the following tasks: It manages main memory, that is, which bytes of memory are utilised by which user application. Memory addresses that have previously been assigned, as well as memory addresses that have not yet been utilised. In multiprogramming, the operating system determines which processes have access to memory and for how long. It allocates memory to a process when it asks it and dealslocates memory when the process terminates or performs an I/O activity.

Processor Management: In a multi-programming environment, the operating system determines the sequence in which programmes access the processor and how much processing time each process gets. This OS operation is known as process scheduling. For processor management, an operating system performs the following tasks.

Maintains a record of the status of processes: A traffic controller is the software that performs this function. Allocates a processor's CPU to a process. When a process is no longer necessary, the processor is de-allocated.

Device Management: An operating system maintains device connectivity using drivers. It manages devices by doing the following tasks. It keeps track of all the devices that are linked to the system. The Input/Output controller is a software that is responsible for every device. Determines which processes have access to a certain device and for how long. Devices are allocated in an effective and efficient manner. When a gadget is no longer needed, it is dealtlocated.

File Management: A file system is divided into directories to facilitate navigation and use. These folders may include additional directories and files. The following file management tasks are performed by an operating system. It maintains track of where information is kept, user access settings, and the condition of each file, among other things... The file system refers to all of these features. Furthermore, the operating system delivers various services to the computer system in some way or another[5]. The operating system delivers various services to users, which may be summarised as follows:

Program Execution: The Operating System is in charge of the execution of all programmes, whether they are user or system applications. The Operating System makes use of many resources to ensure that all sorts of functions execute smoothly.

Processing Input/Output Operations: The Operating System is in charge of handling all types of inputs, such as those from the keyboard, mouse, and desktop. The operating system handles all interfaces for all types of Inputs and Outputs in the most suitable way. For example, the nature of all sorts of peripheral devices, such as mice or keyboards, differs, and the Operating System is responsible for processing data between them.

File System Manipulation: The Operating System is in charge of making choices about the storage of all forms of data or files, such as floppy disks/hard disks/pen drives, and so on. The Operating System determines how data is processed and stored.

Error Detection and Handling: The Operating System is in charge of detecting any sort of error or bug that may arise while doing any operation. A well-secured operating system may also operate as a countermeasure to avoid any kind of intrusion to the Computer System from any external source and perhaps handle them.

Resource Allocation: The Operating System guarantees that all available resources are utilised properly by determining which resources are to be used by whom and for how long. The Operating System makes all of the choices.

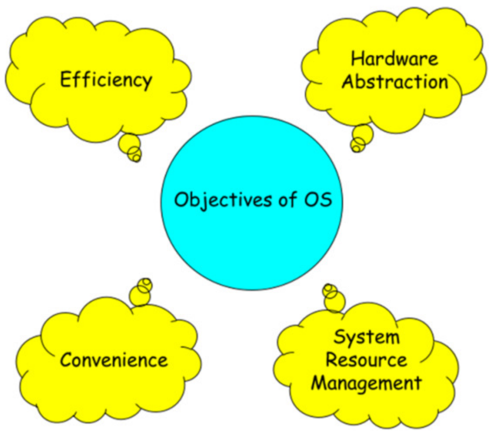
Accounting: The Operating System keeps track of all the functions that are active in the computer system at any one moment. The Operating System records all information, such as the sorts of mistakes that happened.

Information and Resource Protection: The Operating System is in charge of making the best use of all the information and resources available on the computer. The operating system must prevent any external resource from interfering with any kind of data or information[6].

Goals of the operating system

The operating system serves as a link between the computer user and the computer hardware. On top of the operating system are all of the apps necessary for your programmes to use the computer hardware. The following are the primary goals of an operating system:

A. Productivity
B. Hardware abstraction
C. Convenience
D. System resource management



Efficiency: The operating system improves manufacturing efficiency. This is due to the fact that system setup takes less time. The operating system performs system activities including distributing resources to processes and resolving disputes between applications and users by default. This saves the user time and produces a more efficient outcome.

Hardware abstraction: The operating system does an excellent job of hiding the computer's internal characteristics. The user may completely use the computer hardware without encountering any complications. The operating system manages the interaction between user applications and computer hardware.

Convenience: Customers would have to deal with the hardware description language without recourse to the pre-configured utility bundles that come with an operating system if there was no operating system. Using a computer would be highly inconvenient as a result. Operating systems enable users to get straight to work on the activities they want to complete without having to deal with the hassle of setting up the system first.

System resource management: The operating system acts as an impartial arbiter. It manages the computer system by guaranteeing equal resource allocation among diverse processes and customers[7].

Evolution of Operating System

Over time, operating systems have changed. Thus, the development of operating systems across time may chart their progression. There are four operating system generations. These can be described as follows:

1. First Generation (1945 - 1955): Plug boards and Vacuum Tubes

The development of digital computers waited until after World War II. At that time, calculation engines with mechanical relays were constructed. Vacuum tubes later took the place of the mechanical relays, which were slow. Despite being massive, these machines were still moving quite slowly. One team of individuals created, assembled, and maintained these early computers. Machine language was used exclusively for programming because there were no other programming languages or operating systems. The issues were all straightforward mathematical calculations. Punch cards were used in the 1950s, which enhanced the computer system. Programs written on chips and read into the system instead of plug boards.

2. Second Generation (1955 - 1965): Batch Systems and Transistors

The invention of transistors paved the way for the creation of commercially viable computers. The mainframes were kept locked up in air-conditioned computer rooms with people to manage them. To cut down on computer waste, the Batch System was developed. In the input room, a tray of jobs was gathered and read into the magnetic tape. The tape was then wrapped again and installed on a tape drive. The first job was then read off the tape and executed by the batch operating system, which was then loaded. On the second tape, the output was recorded. The input and output tapes were taken out, and the output tape was produced, after the entire batch had been processed[8].

3. Third Generation (1965 - 1980): Multiprogramming and Integrated Circuits

Systems from the 1960s were batch processing systems as well, but by performing several processes concurrently, they were able to make better use of the computer's resources. In order to keep several jobs moving forward while maintaining the utilization of peripheral devices, operating system designers created the notion of multiprogramming, in which multiple jobs exist in main memory at once. A processor is moved from job to job as needed. For instance, on a system without multiprogramming, the CPU would just sit idle while the current process was pausing to wait for another I/O transaction to finish. Memory was divided into numerous sections, each of which contained a separate task as a response to the difficulty that developed. The CPU might be used by one job while it waited for I/O to finish. The spooling method was a key component of the third-generation operating system.

Spooling is the process of placing a high-speed device such as a disk between a program that is running and a low-speed device that is involved in the program's input and output. For example, outputs are written to the disc rather than straight to a printer. When the printer is made available, applications can run more quickly and other program can be started sooner. This allows for the possibility of printing the results. Be aware that the spooling technique is quite similar to the process of winding thread onto a spool so that it can be unwound as necessary. This generation also included the time-sharing approach, a type of multiprogramming technique in which each user has a directly linked (or online) terminal. The computer system must react to user requests rapidly since the user is there and interacts with the computer; otherwise, user productivity can decrease. Systems for timesharing were created to multi-program many interactive users at once.

4. Fourth Generation (1980 - Present): Personal Computers

Large-scale integrated circuits allowed for the widespread use of computers. Departments had been able to purchase their own computers thanks to minicomputers, but personal computers offered everyone access to computational power. As a result, operating systems had to be developed very differently. It may no longer be necessary to have a highly skilled specialist to run an IBM 360; anyone might now utilise computers directly. There are still two operating systems from this time period. Only the most fundamental aspects of a contemporary operating system were included in MS-DOS, an operating system that IBM licenced from Microsoft. On the other hand, the Apple Macintosh's operating system was a successful attempt to design a user-friendly system that could be easily handled with a mouse through the use of a graphical user interface. It has long been restricted to specialised uses like graphic design, publishing, and advertising as a result of its initial high cost. MS-DOS and UNIX are currently attempting to overtake Apple's lead in the graphical user interface market. Windows 95, created by Microsoft, is nearly as user-friendly as Apple's operating system. While graphical system administration tools are being offered by UNIX-based workstation makers like Hewlett Packard and IBM with their UNIX variations in an effort to catch up[9].
Computer networks expanded during this generation's development, enabling the sharing of resources from printers and discs in basic systems to computing power in more complex ones. While MS-DOS and Apple system can only be utilised if the user is physically present at the computer, UNIX allowing users on other computers to remotely access them. This describes the

functions that UNIX is used for. UNIX computers are utilised for complex calculations and database activities, while MS-DOS and Apple systems are suitable for word processing and spreadsheets. It is significant to highlight two additional high power systems in this context. The very secure VMS system from Digital was created initially for the VAX line of computers. Another extremely safe system is Windows NT from Microsoft, which only shares the user interface with Windows 95 or Windows 3.x. However, Windows NT is only compatible with roughly 4 processors per machine. In contrast to the Windows NT system, which is less popular because of its restrictions on remote utilization and processing capacity, the VMS system has a significant user base because of its age. The percentage of UNIX, VMS, and Windows NT systems sold by Digital, which sells all three platforms for its Alpha line of workstations, is roughly 60%, 30%, and 10%, respectively. The relative user-unfriendliness of UNIX in this range of powerful machines is balanced out by the fact that most users today use desktop computers running Microsoft's Windows or Apple's Macintosh, whereas only a tiny number of professionals deal with these systems.

Architecture of Operating Systems

The operating system offers a setting in which users can run computer programs. The operating systems that come with the machines you purchase include Windows, Linux, and macOS for personal computers, z/OS and z/VM for mainframe computers, and iOS and Android for mobile devices. Each of the four primary components of an operating system's architecture in more depth.

1. Hardware
2. Kernel
3. Shell
4. Application

1. Hardware:

The components of the hardware are the memory, CPU, arithmetic-logic unit, various bulk storage devices, I/O, peripheral devices, and other tangible objects.

2. Kernel:

The system that creates of an OS is referred to as the kernel. The OS's kernel's primary function is to choose which component loads into the processor first and which stays in the main memory. Simply put, the kernel determines which operation should be held in main memory for later execution and which one should be assigned to the processor to run immediately. All of this is taking place as a result of the kernel, which is present in memory and is as little as feasible. The kernel is in charge of offering all the fundamental functions needed by other operating system components and user applications.

3. Shell:

The shell is a component of the software that performs kernel functions and is positioned between the user and the kernel. Thus, the shell serves as an interpreter to translate user commands into machine code. Command-line shells and graphics shells are the two types of shells seen in various operating systems. While graphics line shells offer a graphical user interface, command-line shells offer a command-line interface. Although both shells are functional, the shells with graphical user interfaces operate more slowly than the shells with command-line interfaces.

4. Application:

An operating system's architecture is essentially the layout of its hardware and software components. The operating system that is most suited for that program or software can be used by users depending on the operations or programs they need to run.

Categories for Architectures of Operating System

A. Monolithic Architectures
B. Layered Architectures
C. Microkernel Architectures
D. Hybrid Architectures

Monolithic Architectures:

Multi-related tasks can be handled by monolithic software. They frequently involve numerous tightly coupled functions and are sophisticated applications. Take a monolithic SaaS application for online shopping as an illustration. It might include a web service, a load balancing, a service that serves up product images from a catalogue, an ordering platform, a payment option, and a delivery element.

1. **Layered Architectures:**

Exclusively monolithic designs became awkward as operating systems grew bigger and more complicated. The multimodal approach to operating systems makes an attempt to solve this problem by classifying components into layers that carry out related tasks. Only the layers directly above and below it can communicate with each other. Higher-level levels receive services from lower-level layers using an interface that conceals how those services are implemented. Example-Windows XP, and LINUX.

2. **Microkernel Architectures:**

In this design, memory management and synchronization are carried out inside the kernel, whereas elements like device management process management, file system interaction, and networking, are carried out outside the kernel. The operating system continues to function even if one or more components fail because the processes inside the kernel have a comparatively high priority and the elements have a high degree of modularity.

3. **Hybrid Architectures:**

The term "hybrid architecture" refers to an architecture that combines all of the previously discussed architectures. As a result, it possesses characteristics of all of those frameworks and is therefore very useful in modern operating systems.

Findings:

A. Different operating system architectures allow us to explain the functionality of different components.
B. I/O management, Process management, error detection, memory management, and peripheral device control are all parts of the operating system.
C. These architectures are categorized according to the component structure and include microkernel, multilayer, monolithic, and hybrid architectures.

Because it incorporates the features of all previous architectures, hybrid architecture is the most effective and practical architecture. Additionally, hybrid architecture is more secure[10].

REFERENCES

[1] F. R. A. Hopgood and A. J. T. Colin, "Introduction to Operating Systems," Math. Gaz., 1973, doi: 10.2307/3615198.

[2] "EVOLUTION OF ANDROID OPERATING SYSTEM: A REVIEW," Asia Pacific J. Contemp. Educ. Commun. Technol., 2018, doi: 10.25275/apjcectv4i1ict2.

[3] Z. N. Chen et al., "Evolution of Cloud Operating System: From Technology to Ecosystem," J. Comput. Sci. Technol., 2017, doi: 10.1007/s11390-017-1717-z.

[4] Z. ZHENG and G. XIAO, "Evolution analysis of a UAV real-time operating system from a network perspective," Chinese J. Aeronaut., 2019, doi: 10.1016/j.cja.2018.04.011.

[5] J. Wang, Y. Li, Y. Tan, Q. Wu, and Q. Wu, "Analysis of open source operating system evolution: A perspective from package dependency network motif," Symmetry (Basel)., 2019, doi: 10.3390/sym11030298.

[6] F. Guan, L. Peng, L. Perneel, and M. Timmerman, "Open source FreeRTOS as a case study in real-time operating system evolution," J. Syst. Softw., 2016, doi: 10.1016/j.jss.2016.04.063.

[7] R. Q. Tang, J. M. Wagner, H. S. Alper, X. Q. Zhao, and F. W. Bai, "Design, Evolution, and Characterization of a Xylose Biosensor in Escherichia coli Using the XylR/ xylO System with an Expanded Operating Range," ACS Synth. Biol., 2020, doi: 10.1021/acssynbio.0c00225.

[8] L. Vichard, F. Harel, A. Ravey, P. Venet, and D. Hissel, "Degradation prediction of PEM fuel cell based on artificial intelligence," Int. J. Hydrogen Energy, 2020, doi: 10.1016/j.ijhydene.2020.03.209.

[9] D. Spinellis, P. Louridas, and M. Kechagia, "The evolution of c programming practices: A study of the unix operating system 1973-2015," in Proceedings - International Conference on Software Engineering, 2016. doi: 10.1145/2884781.2884799.

[10] J. Wang, K. Zhang, X. Sun, Y. Tan, Q. Wu, and Q. Wu, "Package network model: A way to capture holistic structural features of open-source operating systems," Symmetry (Basel)., 2019, doi: 10.3390/sym11020172.

CHAPTER 3
TYPES OF OPERATING SYSTEMS

Dr.Swapna M, Associate Professor
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- m.swapna@presidencyuniversity.in

The functions of an operating system include processing, file management, and resource management. It serves as a user interface for the computer system. There are numerous varieties of operating systems available. The benefits and drawbacks of several types of OS are covered in this section.

1. Batch Operating Systems
2. Distributed Operating Systems
3. Multiprocessing Operating Systems
4. Multiprogramming Operating Systems
5. Network Operating Systems
6. Time-Sharing Operating Systems
7. Real-Time Operating Systems

Batch Operating Systems

Computers were very huge devices that were controlled by a console. Tape drives, punch cards, and line printers were often utilized for output and tape drivers or card readers for inputs. The system had no direct user interface, and jobs were executed in a batch system. Users must separately prepare each job to be executed on these systems, which are also referred to as batched operating systems. In the 1950s and 1960s, there were no sophisticated operating systems, intricate computer architectures, or supplementary memory devices. Punched cards or magnetic tapes were utilized as input and output, and enormous mainframe computers were employed to handle the data. At the time, a scarcity of hard discs was the main issue. The earliest Single-Stream batch processing systems were introduced by General Motors Research Laboratories (GMRL) in the early 1950s. Data was delivered in groups or batches, and only one job could be completed at once. The setup time problem is resolved by the batch operating system.

Batch processing was very common in the 1970s. The jobs were finished using batch processing. A mainframe, which was a single computer, was once in use. When using batch OS, users don't communicate with the computer directly. Using an offline device like a punch card, each user prepares their work and submits it to the computer operator. Jobs with similar criteria are combined and processed as a group to speed up processing. The operator sorts the programs into batches according to their requirements after receiving the programs from the programmers. The batch operating system was used to organize jobs that carry out similar tasks. These work groups are processed in batches and executed simultaneously[1]. A computer running this operating system can carry out the following batch processing tasks as shown in the Figure:

A. A job is an integrated collection of statistics, commands, and programs that has been pre-programmed.

B. First come, first served, means that the orders are processed in the order that they are received.
C. These tasks run automatically without any interaction from the user because they are saved in memory.
D. After a job has been successfully performed, the OS releases memory.

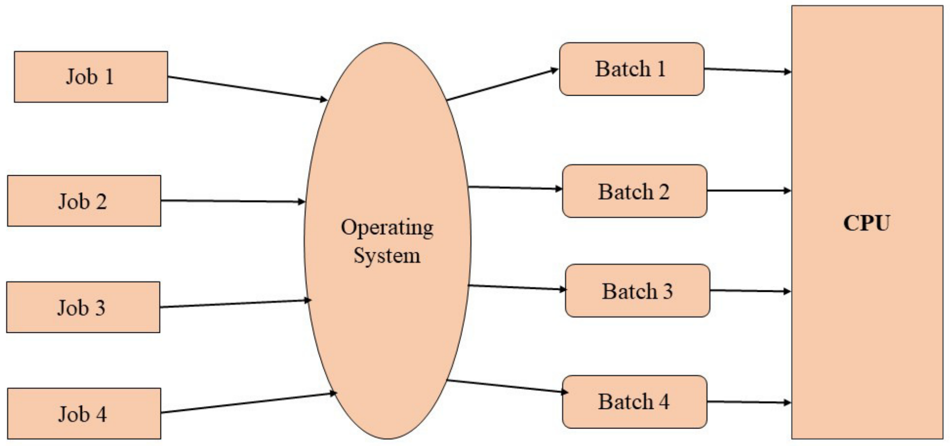


Figure 3.1: Batch operating system

Types of Batch Operating System

The two categories into which the batch OS can be classified are described below.

Simple Batched System

In a straightforward batch operating system, the user did not specifically communicate with the computer system during job execution. However, the user has to set up a task with the programme, control information, and information about the project's nature on control cards. The job was then given to the computer operator, who typically used a punch card, to complete. In the event of a program error, the program's output comprised results, registers, and memory dumps. After a period of time which may be days, hours, or minutes the production became visible.Transferring control from one task to another was its primary function. To increase processing speed, jobs with comparable criteria were grouped together and executed via the processor. The program made use of the operators to construct batches with related requirements. When new batches are available, the computer runs them one at a time. Typically, this system scans a series of jobs, each of which has a control controller and predetermined work responsibilities[2].

Multi-Programmed Batched System:

Numerous jobs that have been read and are waiting to run on disc are handled by spooling. To maximize CPU usage, the operating system can select which work to execute next from a pool of jobs stored on a disc. Directly received jobs from magnetic tape or cards cannot be run in a different order. Because they are executed on a first-come, first-served basis, jobs operate sequentially. Job scheduling is made possible, just like with a disc, when numerous jobs are stored on a direct access device. The ability to multi-program is a crucial component of task scheduling. Spooling and offline procedures have their restrictions for overlapped I/O. In most cases, a single user could not continuously manage all of the input/output devices and CPU purchases.

To increase CPU usage, jobs in the multi-programmed batched system are grouped together so that the CPU only runs one work at a time. The operating system keeps a number of tasks running in memory at once. One job is chosen by the operating system, and it starts running in memory. Last but not least, the job needs to wait for a task to finish, such mounting a tape on an I/O operation. Avoid sitting still in a multiprogramming system since the operating system might move to another task. The CPU is restored after the completion of the job that is currently in the wait state[3].

Advantages:

A. Only the batch processing operating system processors have a good idea of how long it will take to complete a task, thus estimating that time is quite difficult.
B. As soon as the current job is finished running, a new one is instantly loaded, which improves system performance and speeds up setup.
C. The single batch processing operating system can be shared by many users.
D. In a multi-programmed batch operating system, the CPU spends relatively little time idle since it is constantly working on a task.
E. Due to the fact that the entire batch is being processed in a single environment, we need to swap contexts less frequently between environments after each process.

Drawbacks:

A. In a single batch processing operating system, the CPU isn't used when a specific I/O operation is required.
B. A CPU cannot execute any other operation in the middle of one task if the I/O device and CPU speeds are out of sync.
C. Important tasks cannot be given a high priority by the user; they all run in order.
D. User must wait until all processes have completed before seeing results.

Due to the fact that some tasks may take a long time to complete and others may need to wait for resources, the execution of tasks can occasionally result in starvation.

Distributed operating systems

An operating system (OS) is essentially a group of programs that manages the hardware resources of a computer and offers standard services to programs running on it. An essential part of a computer system's system software is the operating system. One of the most significant types of operating systems is the distributed operating system. Distributed systems employ multiple central processors to support numerous real-time applications and numerous users. Jobs related to data processing are therefore divided among the processors. Various communication channels are used by processors to exchange information (like high-speed buses or telephone lines). These are referred to as dispersed systems or loosely connected systems. The size and purpose of the processors in this system can differ. They go by the names sites, nodes, computers, etc[4].

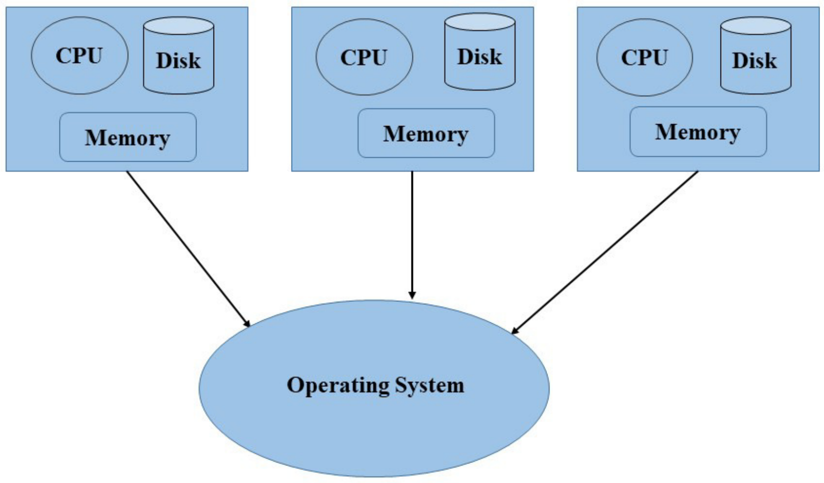


Figure 3.2: Distributed operating system

The various computers, nodes, and sites that make up this operating system are connected by LAN/WAN links. It supports a wide range of real-time goods and diverse users, and it allows the dispersion of whole systems across a few center processors. Distributed operating systems allow users to abstract away from physical machines while sharing compute resources and I/O files.

Distributed Operating System Types

Different kinds of distributed operating systems exist. Here are a few of them:

1. Client-Server Systems
2. Peer-to-Peer Systems
3. Middleware
4. Three-tier
5. N-tier

1. Client-Server Architecture

In this kind of system, the client must first request a resource, which the server then provides. A server may serve numerous clients simultaneously when user connects to it.

Tightly Connected Operating Systems is another name for Client-Server Systems. Multiprocessors and homogeneous multicomputer are the main targets of this technology. Due to their role as a centralized server that grants all requests made by client systems, client-server systems perform this purpose[4].

There are two types of ever systems:

1.1.Computer Server System

The interface is enabled by this system, and the client then sends its own requests to be executed as an action. After finishing the activity, it sends a response and gives the client the outcome.

1.2. File Server System

It provides users with a file system interface so they may do operations like creating, deleting, and updating files.

2. A peer-to-peer network

Nodes have a big part in this system. The nodes each receive an equal share of the work. These nodes can also exchange data and resources as needed. To reconnect, they need some sort of network. The peer-to-peer system is referred to as a "Loosely Coupled System." This idea is employed because computer network applications require a sizable number of processors that do not share memory or clocks. Each CPU contains its own local storage, and they interact with one another via a variety of channels, such as high-speed buses and telephone lines[5].

3. Middleware

The compatibility of all apps running on various operating systems is made possible by middleware. By utilizing these services, those programs are able to transport any type of data amongst one another.

4. Three-tier

Development is made easier because the customer's information is saved in the intermediate tier rather than the client. Online applications are where this type of architecture is most frequently found.

5. N-tier

N-tier systems are utilized when a service or application has to send messages to other enterprise solutions over the network.

Distributed operating system applications

The following are some typical uses for distributed operating systems:

A. Using the Internet
B. Air traffic management
C. Reservation management systems for airlines
D. Networks for communications
E. Computer clusters
F. Rendering data
G. Using a grid

Distributed OS benefits

A. The system is under less load.
B. One system failing won't have an impact on the other.
C. Calculations are simple because of the system's burden sharing.

D. Depending on the needs, the system's size can be adjusted.

Distributed OS disadvantages

A. The price of setup is higher.
B. The entire system will be impacted if the primary system fails.
C. Programming is difficult.

Multiprocessing Operating Systems

The symmetric multiprocessing model serves as the foundation for multiprocessing systems, which execute identical copies of the operating system on each processor and interact with one another. In this system, each CPU is tasked with a certain duty. The system is in command of a master processor. In this system, a master-slave relationship is established. Because the processors can share controllers, power supply, and other equipment, these systems can be less expensive than those with a single processor. The ability to complete more work in less time is the main benefit of multiprocessor systems. Additionally, multiprocessor systems show to be more reliable when one processor fails. In this case, the multiprocessor system won't shut down; rather, it will just slow down. Examples UNIX[5].

Types of Multiprocessing Operating Systems:

1. **Asymmetrical Multiprocessing Operating System**

2. **Symmetrical Multiprocessing Operating System**

1. Asymmetrical Multiprocessing Operating System

This entails one processor acting as the master and the rest as the slaves. The slave processors are given tasks by the master processor that are ready for execution. The master processor maintains a ready queue where processes are delivered to subordinate processors for execution. A scheduler created by the master processor distributes operations to other processor in an asymmetric system.

Advantages

A. A master CPU distributes tasks to other processors, greatly reducing the possibility of conflicts.
B. The entire execution procedure is quicker.
C. Easier to design and handle.

Disadvantages

A. All processors might not always receive an equal distribution of jobs. It's possible that some processors are overworked.
B. The system crashes if the processor in charge of a particular task malfunctions.
C. Common buses, memory, peripherals, and clocks for computers.

2. **Symmetrical Multiprocessing Operating System**

All of the processors have their own process schedulers, and there are no master-slave relationships among them. All of the processes that are ready to run are kept in a global queue.

Processes are distributed to all processors from this global queue. Two processors may occasionally receive the same process. This disagreement is resolved via problem locking. Each processor has a local scheduler tied to it in order to maintain balance and guarantee optimal CPU utilisation. As more processes run simultaneously, the throughput of an asymmetric multiprocessing operating system increases.

Advantages

A. It is resilient to errors. As opposed to asymmetric multiprocessing, when one process fails, the entire system does not crash.
B. It is a trustworthy system.
C. The number of processes completed in a given amount of time also rises as a result of the ability to run the operations on several processors.

Disadvantages

A. It Has An Intricate Layout.
B. Difficult To Manage
C. Much More Expensive Than Asymmetric Multiprocessing.

Benefits of Multiprocessing Operating Systems:

The following are some benefits of multiprocessor systems:

A. More processes can run concurrently in parallel if there are many processors active at once. As a result, the system's throughput will rise.
B. Systems with many processors are more reliable. The fact that there are several processors means that even if one of them fails, the system won't shut down. Even though the system will slow down if this occurs, it will still function.
C. A multiprocessor system uses less electricity than a single processor system. This is due to the heavy demand placed on a single processor in single processor systems, which must execute numerous processes. The load on every processor will be relatively lower in systems with multiple processors since there are more processors available to carry out the activities. As a result, less electricity will be used[6].

Multiprocessing Operating System Drawbacks:

A. The complexity of a multiprocessor is present in both its hardware and software forms.
B. Due to its expansive architecture, it is more expensive.
C. Due of its sharing nature, multiprocessor operating systems present a challenging issue for scheduling activities.
D. Because multiprocessor systems share their memory with other resources, they require a lot of memory.
E. Its speed may decrease if one of its processors fails.
F. When the processor receives the message and takes the required action, there is a longer delay.
G. It has significant skew and determinism-related challenges.

It requires context switching, which could affect how well it performs.

Multiprogramming Operating Systems

One or more programs can be put into the main memory of the multiprogramming system in order to be executed. Only one program or process can use the CPU to execute instructions, and all other programs must wait for their turn. The primary objective of using a multiprogramming system is to solve the problem of underutilization of the CPU and main memory. Multiple users can finish their jobs simultaneously and conserve some main memory in the multiprogramming operating system[7]. The CPU may allot time to other programs while one program is running I/O activities as shown in the Figure.

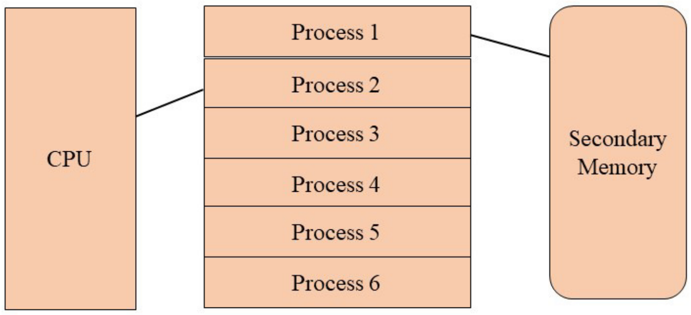


Figure 3.3: Multiprogramming operating system

While one application waits for an I/O transaction while another is continually ready to use the processor, many programs may share CPU time. Multiple jobs may run on the processor at once, albeit not all tasks are performed concurrently. Parts of other processes may be finished first, then another segment, and so on. The operating system must make sure CPU resources are properly allotted to each program and reassigned as resources become present in order for all the programs to run. The OS does this by using context switching, which guarantees that a program's state is maintained in memory and is readily available as the OS switches between programs and CPU assignments. The overall goal of a multiprogramming operating system is to keep the CPU active until each task is fully completed. As a result, a computer with a single processor can run multiple programs at once and never go into standby mode.

Types of Operating System:

Two primary categories of multiprogramming operating systems are described below, one for each:

1. Multitasking Operating System
2. Multiuser Operating System

1. Multitasking Operating System

An interface for running many program tasks simultaneously by a single user on a single computer system is provided by a multitasking operating system. Any editing work, for instance, can be carried out while other programs are running simultaneously. Another example is the ability to simultaneously open Power Point and Gmail.

2. Multiuser Operating System

A multiuser operating system enables numerous people to share a highly centralized computer from various terminals. A fixed amount of CPU time on the main computer is assigned to each terminal, and the operating system swiftly switches between them to do this. Because of how quickly the operating system switches between terminals, it looks as though each user has continuous connection to the main computer. The more users there are on a systems like this, the more likely it is that the response time of the main computer will be more apparent[8].

Benefits of Multiprogramming Operating Systems:

A. It offers a slower response time.
B. Running several tasks at once in one application could be helpful.
C. It aids in improving the computer's overall job throughput.
D. The multiprogramming system can be used by multiple people at once.
E. Jobs that are completed swiftly are those that are completed promptly.
F. It might aid in reducing turnaround time for urgent tasks.
G. It increases CPU efficiency and is constantly active.
H. The resources are widely used.

Drawbacks of Multiprogramming Operating Systems:

A. Long-running CPU operations occasionally need to wait for other, typically quicker, tasks to complete.
B. Multiprogramming makes it difficult to manage numerous tasks at once.
C. Because scheduling is so intricate, multiprogramming operating systems is not a simple task.
D. Operating systems with multiple programs must use CPU scheduling.
E. Memory management ought to be quite effective.
F. There cannot be any communication between a program and the user while it is running.

Multitasking Operating Systems

An advanced computer system uses the term "multitasking." It is a logical progression from the multiprogramming system, which enables many applications to operate at once. A user can perform several computer operations concurrently thanks to multitasking in an operating system. Many tasks are processes that use common processing resources, such a CPU. The operating system keeps track of your position in each of these jobs so that you can transition between them without losing any data[9].

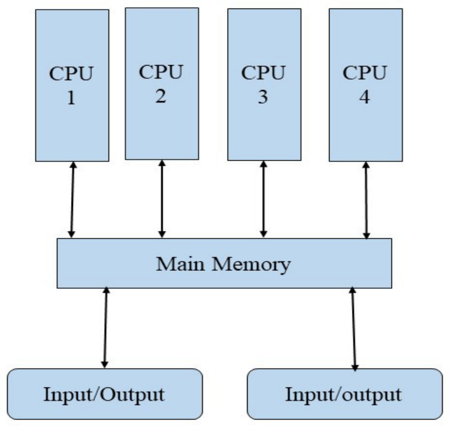


Figure 3.4: Multitasking operating system

Although the early operating systems allowed the simultaneous use of many programs, multitasking was not fully supported. As a consequence, a single bit of software may utilize the computer's entire CPU to finish a task. Because fundamental operating system features like file copying were blocked, the user was also unable to perform additional tasks like opening and closing windows. Multiple programs can run concurrently without interacting with one another because to the robust multitasking offered by modern operating systems. Additionally, a lot of OS tasks can run at once.

Multitasking Operating Systems Types

Multitasking usually comes in two flavors. These are listed below:

1. Preemptive Multitasking
2. Cooperative Multitasking

1. Preemptive Multitasking

A computer operating system is given the particular task of preemptive multitasking. It determines how much time one task spends using the operating system before giving that task to another. It is called "preemptive" because the operating system manages the entire procedure.Desktop operating systems employ proactive multitasking. The first operating system to employ this technique of multitasking was UNIX. The first versions of Windows to implement preemptive multitasking were Windows NT and Windows 95. The Macintosh gained active multitasking in OS X. When it's time for another program to take over the CPU, this operating system tells the running processes.

2. Cooperative Multitasking

Cooperative multitasking is referred to as "Non-Preemptive Multitasking." Running the current work while releasing the CPU to let another process execute is the basic goal of cooperative multitasking. Using task YIELD (), this task is completed. Context-switch is conducted upon calling the task YIELD () method. MacOS and Windows both employed collaborative multitasking. When a message is received, a Windows program will execute a brief unit of work before turning the CPU over to the operating system and waiting for another message. As long as all applications were bug-free and written with consideration for other programs, it functioned flawlessly.

Benefits of Multitasking Operating System:

1. This OS is better adapted to accommodate multiple concurrent users, and numerous apps can run without the system becoming sluggish.
2. The best virtual memory system is found in multitasking operating systems. Because of virtual memory, no program requires a lengthy wait time to complete its activities; if this issue develops, those apps are moved to virtual memory.
3. Operating systems that support many tasks provide users more freedom, which makes them happy. Additionally, each user can operate one or more software concurrently.

4. Multitasking operating systems have a clearly defined memory management system. As a result, the operating system does not permit unnecessary apps to consume RAM.
5. Each task is allocated a set amount of time to prevent waiting for the CPU.
6. Under a multitasking operating system, background processes can work more effectively. Although they support the efficient operation of other applications like antivirus software, firewalls, and others, the majority of users are not aware of these background processes.
7. A multitasking operating system (OS) can control CPUs, RAM, hard drives, I/O devices, and other computer resources.
8. Users have the option to run several program at once, including games, a web browser, Microsoft PowerPoint, Excel, and other utilities.

Drawbacks of Multitasking Operating System:

1. The systems may run programs slowly, and their response times may lengthen when processing numerous programs, due to the processors' poor speed. It will take more computational power to overcome this obstacle.
2. Multiple apps running at once can have an adverse effect on the computer's performance since it causes the main RAM to become overloaded. Since the CPU is unable to provide separate times for each program, reaction time increases. The main reason of this issue is the use of RAM with low capacity. As a result, the RAM's capacity can be raised to satisfy the needs.
3. Because multiple processors must work simultaneously to complete each task in a multitasking context, the CPU produces more heat.

Multiusers Operating Systems

An operating system that allows several users to connect to a single system running a single operating system is known as a multi-user operating system. These systems, which are frequently rather complicated, are responsible for managing the duties needed by the many users who are connected to them. Users often use terminals, computers, and other system devices like printers that are networked to the system. Because each users access the same operating system from various machines, a multi-user operating system differs from an interconnected single-user operating system.

The primary reason for creating a multi-user operating system is so that it can be used for batch processing and time-sharing on mainframe systems.Large corporations, the public sector, colleges and universities, as well as servers running Windows Server or Ubuntu Server, frequently employ this multi-user operating system nowadays. These servers allow several users to simultaneously access the hardware, kernel, and operating system. It typically manages the memory and processing for other programs that are already executing, locates and makes use of system hardware, and effectively manages users to interact and data demands. Because multiple users depend on the operating system to run properly at once, it is particularly crucial for multi-user operating systems[10].

Components of the Multiusers Operating System:

The multi-user operating system's components are:

1. Memory:

Main memory makes up memory (RAM). The operating system's main memory is considered crucial because it limits the number of concurrently running apps. Every program that is run must be copied from physical storage since the system can rectify data that is in the main memory. The various kinds of physical storage.

1. **Hard drive:** How many programs may be executed simultaneously depends on the hard disc, which can store a significant quantity of data.
2. **Floppy Disks:** Although more affordable, it is still potable.
3. **Optical Disks:** Lasers are used to read and write data on optical drives. They are portable and can contain a lot of data.
4. **Tapes:** They are pricey but have a substantial capacity amount of information.

2. Kernel:

This part may communicate directly with the hardware of the computer system since it is incorporated into the primary memory of the device. The kernel component is used by the multi-user operating system at a low level, and it is written in a low-level language.

3. Processor:

The central processing unit (CPU) is the computer's brain (CPU).

4. Device handler:

The main objective of the device handler is to fulfil all responses from the entire device waiting queue. The I/O request block from the queue side is the first thing the device handler discards when operating in continuous cycle mode. The handler is based on the idea of a queue where the first in, first out (FIFO), rule is followed.

5. Spooler:

Line-based simultaneous peripheral output. All computer activities are carried out by the spooler, which outputs the findings at the same time.

Types of Multi-user Operating Systems:

The multi-user operating systems is of the following types:

1. Distributed System
2. Time sliced system
3. Multiprocessor system

1. Distributed System

In this, a number of different components are dispersed across numerous computers, interacting and working together as one cohesive system. To put it another way, several computers are managed such that they behave like a single computer. Distributed computing is another name for distributed systems.

2. Time sliced system

In such a systems, the CPU allots a certain amount of time for each user's task. The Scheduler, an internal system, makes the choice of which job to execute next. Based on the priority cycle, the scheduler selects and performs the necessary tasks.

3. **Multiprocessor system**

Such a system uses numerous processors, which enhances the performance of the system as a whole. The other processors are in charge of finishing the duties left unfinished by the failed processor, preventing system failure.

Benefits of a Multi User Operating System

1. A multi-user operating system is highly helpful in workplaces or libraries since it can effectively handle printing tasks.
2. On a single computer system, several users can access the same copy of a document. For instance, if a PPT file is kept on a single computer, additional users can see it on different terminals.
3. The ticket reservation system employs a multi-user operating system.
4. Some airline functions also make use of multi-user operating systems.
5. If one computer in a network system breaks, the overall structure does not come to a stop.
6. Because manufacturing records are housed in a single system and personnel are not restricted to a single computer, all firm managers can search manufacturing data using the multi-user operating system as well.

Multi-user Operating System's drawbacks

When a virus infects one network of computers using the multi-user operating system, it opens the door for the virus to spread to the rest of the machines in the network.

Data visibility: As all of the information in computers is shared in public, privacy of information and data becomes a worry.

Network Operating Systems

An operating system (OS) is essentially a group of programs that manages the hardware resources of a computer and offers standard services to programs running on it. An essential part of a computer system's system software is the operating system. One of the most significant types of operating systems is the network operating system. A server's Network Operating System allows for the management of data, customers, groups, safety, programs, and other networking tasks. The fundamental function of a network operating system is to enable shared file and printer access among numerous computers connected to a network, often a local area network (LAN), a private network, or to other networks. Linux, Microsoft Windows Server 2003, UNIX, and Mac are a few examples of network operating systems.

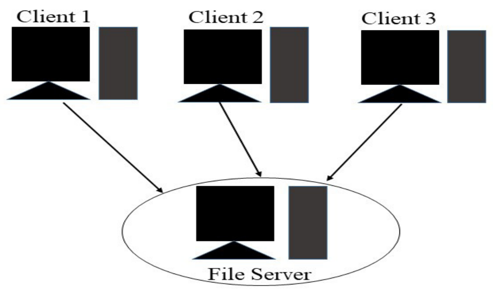


Figure 3.5: Network operating systems

It enables shared hardware between computers, such as discs, printers, etc. The Network Operating System is installed on top of each of the individual devices that make up the Network, each of which has its own operating system. Individual computers must log into another computer using the correct password since they each have their own Operating System that allows them to access resource from other computers. Additionally, this functionality prevents process migration and prevents communication between processes operating on different machines. The standard network protocol is called the transmission control protocol.

Types of Network Operating System

There are primarily two varieties of Network Operating System.

1. Peer-to-Peer
2. Client-Server

1. **Peer-to-Peer:**

Operating systems for peer-to-peer networks let small networks with limited resources share files and resources. LANs typically use peer-to-peer network operating systems.

Benefits of the Peer-to-Peer Network Operating System:

1. Simple to install and configure
2. The setup is inexpensive.
3. It is not necessary to use any specific software.
4. It is quick and simple to share resources and information.

Drawbacks of the Peer-to-Peer Network Operating System:

1. When sharing resources, the performance of autonomous computers might not be that great.
2. No centralized management exists.
3. Less security exists.
4. It doesn't have any backup features.
5. No centralized storage system exists.

2. Client-Server

Users of client-server network operating systems can access resources via the main server. The cost of implementing and maintaining this NOS is too high. This operating system is suitable for large networks that offer a variety of services.

Client-Server Network Operating System Benefits:

1. Due to centralized data protection, this network is more secure than the peer-to-peer network system.
2. Due to the separation of labor between clients and the server, network traffic is reduced.

3. Large and contemporary organizations value it because it distributes processing and storage across a big area.
4. The Client-Server Network solution enables remote access to the server across several platforms.

Client-Server Network Operating Systems' Drawbacks:

1. Safety and reliability are crucial concerns in client-server networks. Consequently, network administration requires trained network administrators.
2. Depending on the privacy, resources, and connectivity, establishing the client-server network may be expensive.

Advantages of Network Operating System:

1. It is simple to add new hardware and technological advancements into the system.
2. Remote access to the server is possible from a variety of places and devices.
3. The central server is quite reliable.
4. The server is in charge of security.

Disadvantages of Network Operating System:

1. The central position is crucial to most activities.
2. Purchase and maintenance of servers are expensive.
3. It requires routine upkeep and updating

Time-Sharing Operating Systems

One kind of an operating system type is a time sharing operating system. In essence, an operating system is a piece of software that serves as a conduit between the user and the hardware of the machine. Additionally, it manages all communications between the hardware and software. It enables the user to conduct multiple tasks simultaneously, giving each one an equal amount of time to complete. Hence, the OS known as time sharing. It is also a development of multiprogramming systems. The goal of multiprogramming systems is to utilize the CPU as much as possible. On the other hand, in this case, the goal is to minimize CPU reaction time.

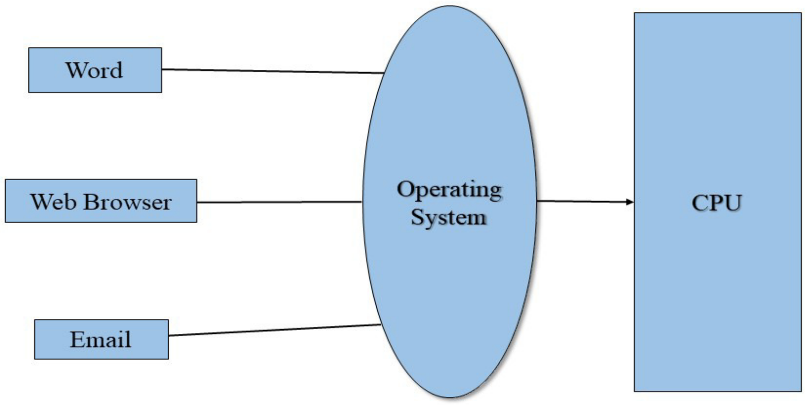


Figure 3.6: Time-Sharing operating system

As an outcome, the system will perform all duties without issue. As a result, it is also known as a multitasking operating system. The CPU is being used simultaneously when several tasks are underway. However, the user feels as though all jobs are running simultaneously because of how quickly tasks are switched between.

Through CPU scheduling and multiprogramming, the operating system implements time sharing. Let's take a hard look at how a time sharing system operates. It goes like this:

1. When the user assigns many tasks, the CPU time is divided among the various processes.
2. Each procedure is given a brief amount of time. Additionally, this time span is extremely brief—between 10 and 100 milliseconds. The terms time slot, time slice, and quantum all refer to this period of time.
3. Assume that the system is executing word, web browser, and email processes. Now imagine that the quantum lasts for four nanoseconds (ns). They will now carry out their plan in the following way.
4. After process word has finished running for 4 nanoseconds, process web browser will begin running for 4 nanoseconds. Furthermore, email runs for 4ns after web browser is finished. This process keeps going till every step is finished.
5. In this manner, only one process is active at once, yet switching between them happens quickly. As a result, the user believes that all processes are active at once.

There are three states during the process of the time sharing operating system.

1. **Active State:**

The CPU is in charge of the user's program. There is only one program offered in this state.

2. **Ready States:**

In the ready state, the user software is prepared to run but is awaiting its turn to access the CPU. At any given time, multiple users may be in the ready state.

3. **Waiting States:**

The user's program is in a waiting state as it awaits an input or output activity. At any given time, multiple users may be in the waiting state.

Time-Sharing Operating Systems Advantages:

Some advantages of the time-sharing operating system include the following.

1. It reacts quickly to commands.
2. Idle CPU time is decreased.
3. A specific time restriction is allocated to each task.
4. Reaction speed is enhanced by a lower probability of program duplication.
5. Easy to use and intuitive.

Time-Sharing Operating Systems Drawbacks:

The time-sharing operating system has a number of drawbacks, including the ones listed below.

1. It makes extensive use of resources.
2. High-quality hardware is necessary.
3. It has a hard time being consistent.
4. A difficulty with user programs and data security and integrity.
5. Probability of data communication problems.

Real Time Operating Systems

An operating system (OS) that ensures real-time applications have a given capability by a certain deadline is known as a real-time operating system (RTOS). RTOSes are intended for time-sensitive devices like microcontrollers and critical systems. Requirements for RTOS processing speed are expressed in milliseconds. Any hesitation in answering could have dire repercussions. Real-time operating systems (RTOSes) perform similar tasks to general-purpose operating systems (GPOSes), such as Linux, macOS, Microsoft Windows, , but are built with a scheduler that can adhere to strict deadlines for certain activities. RTOSes are frequently used in embedded systems, which are made up of hardware and software and are created for a particular purpose. Embedded systems can also serve as part of a larger system. Embedded systems frequently operate in real-time settings and make use of a real-time operating system to interconnect with the hardware. Multiple processes can be handled simultaneously by RTOSes, which guarantees that they respond to events in a predictable amount of time. In an RTOS, processing takes place within predetermined time limits while keeping track of job priority. Task priority can also be modified by an RTOS. Event-driven systems frequently move between activities based on priority. Example: VxWorks, MTS, QNX, Lynx etc.

Types of Real time Operating System

1. Hard Real Time
2. Soft Real Time
3. Firm Real Time

1. Hard Real Time:

The deadline is treated very tightly in Hard RTOS, which means that a given job must begin executing at the scheduled time and must be finished within the allotted time period. Examples: aircraft systems and critical care medical systems.

2. Soft Real Time:

These RTOS must likewise adhere to the deadlines. While missing a deadline may not have a significant effect, it can have unintended consequences, such as a significant decline in product quality. Example: Livestock price quotation System and Online Transaction system.

3. Firm Real Time:

The operating system tolerates some delays from the soft real-time RTOS. When using this kind of RTOS, each task has a deadline that must be met, but a brief delay is permitted. So, this kind of RTOS handles deadlines gently. Examples: many multimedia application kinds.

Real-time Operating Systems Benefits:

1. Real-time operating systems make it simple to design, create, and run real-time applications.
2. Because the real-time working structures are so small, they take up substantially less memory.
3. Maximum system and device utilization in a real-time operating system.
4. Applications that are now running should receive more attention than those that are waiting in line.
5. Since RTOS applications are compact, they can also be embedded systems in other industries, including transportation.
6. These systems don't include any errors.
7. These systems are the ones that manage memory allocation the best.
8. The RTOS system can conduct a small number of activities concurrently, and it focuses solely on applications that have errors so that it can escape them.

Real-time Operating Systems Drawbacks:

1. The system known as RTOS focuses on a small number of tasks. As a result, these systems find it extremely difficult to multitask.
2. The RTOS needs specific drivers in order to respond quickly to interruption signals, which aids in maintaining its pace.
3. RTOS consumes a lot of resources, which drives up the cost of this system.
4. Low priority tasks must wait a lengthy period while the RTOS retains the integrity of the program that is being executed.
5. Real-time operating systems perform the least amount of task switching.
6. It employs complicated algorithms that are challenging to comprehend.

REFERENCES

[1] A. Singh, “Types of Operating Systems - GeeksforGeeks,” Geeks for Geeks, 2018.

[2] S. Xuan, D. Man, W. Yang, W. Wang, J. Zhao, and M. Yu, “Identification of unknown operating system type of Internet of Things terminal device based on RIPPER,” International Journal of Distributed Sensor Networks. 2018. doi: 10.1177/1550147718806707.

[3] U. Cristian, “Types of operating system kernels,” in Instrumentul Bibliometric National, 2020.

[4] P. Sahni and N. Sharma, “File System,” J. Adv. Res. Comput. Sci. Eng. (ISSN 2456-3552), 2014, doi: 10.53555/nncse.v1i4.513.

[5] K. Divyap and S. Venkata Krishnakumar, “COMPARATIVE ANALYSIS OF SMART PHONE OPERATING SYSTEMS ANDROID, APPLE iOS AND WINDOWS,” Int. J. Sci. Eng. Appl. Sci., 2016.

[6] M. A.Ismail, H. Aboelseoud M, and M. B. Senousy, “An Investigation into Access Control in Various Types of Operating Systems,” Int. J. Comput. Appl., 2014, doi: 10.5120/17218-7454.

[7] J. Yang and C. Hawblitzel, “Safe to the Last Instruction: Automated verification of a type-safe operating system,” in ACM SIGPLAN Notices, 2010. doi:

10.1145/1809028.1806610.

[8] I. Irianto, A. Afrisawati, S. Sudarmin, and J. Eska, “IMPLEMENTASI PERAKITAN DAN INSTALASI SISTEM OPERASI WINDOWS DAN LINUX,” Jurdimas (Jurnal Pengabdi. Kpd. Masyarakat) R., 2018, doi: 10.33330/jurdimas.v1i1.386.

[9] D. P. Van, T. Fujiwara, B. L. Tho, P. P. S. Toan, and G. H. Minh, “A review of anaerobic digestion systems for biodegradable waste: Configurations, operating parameters, and current trends,” Environmental Engineering Research. 2020. doi: 10.4491/eer.2018.334.

[10] J. Martin et al., “Handoff All Your Privacy – A Review of Apple’s Bluetooth Low Energy Continuity Protocol,” Proc. Priv. Enhancing Technol., 2019, doi: 10.2478/popets-2019-0057.

CHAPTER 4

COMPUTER SYSTEM MANAGEMENT

Mr. Sanjeev P Kaulgud, Assistant Professor
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- sanjeevkaulgud@presidencyuniversity.in

Computer System Management: Concurrent execution of CPUs and devices vying for memory cycles over a common bus connecting one or more CPUs and device controllers.

Operation of a Computer System

1. I/O devices and the CPU can run concurrently
2. Each device controller is in charge of a specific device type
3. I/O devices and the CPU can run concurrently Common Interrupt Functions
4. Interrupts normally pass control to the interrupt service routine through the interrupt vector, which includes the addresses of all service routines.
5. Interrupt architecture must store the address of the interrupted instruction.
6. A trap is a software-generated interrupt that may be produced by a mistake or by a user request.

Operating system is based on interrupts

The operating system maintains the state of the CPU by storing registers and the programme counter. It determines the sort of interruption that has occurred:

Polling

Vectored interrupt system

For each form of interrupt, separate code segments decide what action should be done.

Timeline Interruption

1. Once I/O begins, control returns to the user application only when I/O is completed.
2. The wait instruction idles the CPU until the next interrupt o The wait loop (contention for memory access) o Only one I/O request is outstanding at a time, no concurrent I/O processing.
3. After I/O begins, control returns to the user application without waiting for I/O completion
4. System call - a request to the operating system to let the user wait for I/O completion.
5. Each I/O device has an entry in the device-status table that indicates its type, address, and state.
6. The operating system indexes into the I/O device table to ascertain device status and to edit table entries to incorporate interrupts.

Structure for Direct Memory Access

1. Used for high-speed I/O devices capable of transmitting data at memory rates.
2. Without CPU interference, the device controller moves data blocks from buffer storage to main memory.

Instead of one interrupt every byte, just one interrupt is produced each block[1].

Structure for Storage:

Main memory - only large storage media that the CPU can directly access.

1. Random access
2. Typically volatile
3. Storage Hierarchy
4. Speed
5. Cost
6. Volatility
7. Caching - transferring information into a quicker storage system; main memory may be considered as a cache for secondary storage

Storage Device Organization

1. Information in use moved from slower to faster storage temporarily
2. Faster storage (cache) evaluated first to verify whether information is there o If it is, information utilised straight from the cache (quick) o If it is not, data copied to cache and used there
3. Cache is smaller than the storage being cached o Cache management is a critical design issue.
4. Cache size and replacement strategy

Architecture of Computer Systems

1. Most systems employ a single general-purpose processor (PDAs to mainframes)
2. Most systems also contain special-purpose processors
3. Multiprocessor systems are becoming more popular and important
4. Also known as parallel systems, tightly-coupled systems o Benefits include:
5. Enhanced throughput
6. Scale economies
7. Improved dependability - graceful degradation or fault tolerance

There are two types:

A. Multiprocessing with Asymmetries
B. Multiprocessing Symmetric

System Calls in Operating System (OS): A system call allows a user software to communicate with the operating system. The software asks a number of services, and the operating system reacts by issuing a sequence of system calls to fulfil the request. A system call may be written in either assembly language or in a high-level language such as C or Pascal. System calls are predefined functions that may be directly invoked by the operating system if a high-level language is employed[2].

Exactly is a System Call: A system call is a means for a computer application to request a service from the operating system's kernel. A system call is a way for applications to communicate with the operating system. A system call is a request from computer software to the kernel of an operating system.The Application Program Interface (API) links the functionalities of the operating system to user applications. It serves as a bridge between the operating system and a process, enabling user-level applications to seek services from the operating system. System calls are the sole way to interact with the kernel system. Any software that consumes resources must utilise system calls.A system call is made when software wants to reach the kernel of an operating system. The system call makes use of an API to expose the services of the operating system to user applications. It is the sole way to get access to the kernel system. System calls are required for any programmes or processes that demand resources for execution because they act as an interface between the operating system and user applications.The examples below show how a system call differs from a user function.

A. To conduct asynchronous processing, a system call function may generate and utilise kernel processes.
B. A system call has more power than a regular subroutine. In the kernel protection domain, a system call with kernel-mode privilege executes.
C. System calls may not utilise shared libraries or any symbols not found in the kernel protection domain.
D. System call code and data are stored in global kernel memory.

Required system calls in the operating system

There are many scenarios in which system calls in the operating system are required. The following are some examples of situations:

A. It is required whenever a file system creates or deletes a file.
B. Network connections need the use of system calls to deliver and receive data packets.
C. To read or write a file, you must use system calls.
D. A system call is required to access hardware devices such as a printer or scanner.
E. System calls are used in the creation and management of new processes.

System Calls Operate

The applications execute in a memory region known as user space. A system call connects to the kernel of the operating system, which operates in kernel space. An application must first get authorization from the kernel before making a system call. This is accomplished by the use of an interrupt request, which stops the current process and passes control to the kernel. If the request is approved, the kernel will carry out the specified action, such as creating or deleting a file. The application gets the kernel's output as input. After receiving the input, the programme continues the operation. When the process is completed, the kernel sends the results to the application and subsequently shifts data in memory from kernel to user space. A basic system call, such as getting the system date and time, may take a few nanoseconds to complete. A more complex system call, such as connecting to a network device, may take a few seconds to complete. To minimise bottlenecks, most operating systems start a separate kernel thread for each system call. Because modern operating systems are multi-threaded, they can process several system calls at the same time[3].

System Call Types

System calls are classified into five categories. These are the following:

A. Process Control

B. File Management
C. Device Management
D. Information Maintenance
E. Communication

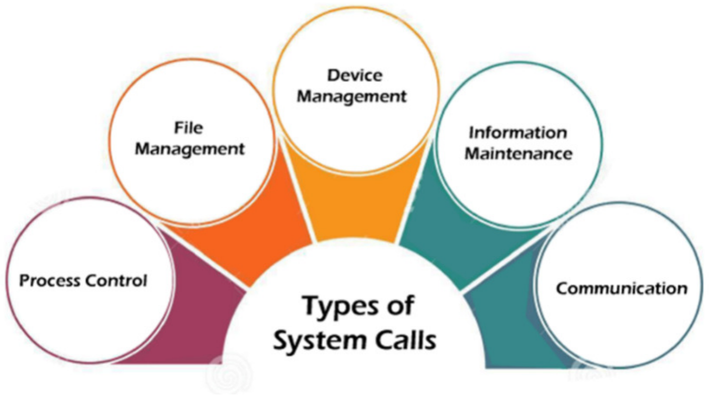


Figure: 4.1: Type of System Calls

The various forms of system calls one by one.

Process Control

Process control is a system call used to guide processes. Process control examples include generating, loading, aborting, terminating the process, and so on.

File Management

File management is a system call used to manage files. Examples of file management include creating files, deleting files, opening, closing, reading, writing, and so on.

Device Management

Device management is a system call used to manage devices. Device management examples include read, device, write, retrieve device characteristics, release device, and so on.

Information Preservation

Information maintenance is a system call that is used to keep information up to date. Receiving system data, setting time or date, getting time or date, setting system data, and so on are some instances of information maintenance.

Communication

A system call used for communication is communication. Some instances of communication include creating and deleting communication links, sending and receiving messages, and so on.

Important System Calls in Operating Systems

wait(): In certain systems, a process must wait for another process to complete before proceeding. When a parent process produces a child process, the parent process's execution is halted until the child process completes its execution. The parent process is automatically halted when a wait () system call is issued. Once the child process has finished its execution, control returns to the parent process[4].

fork(): The fork system function in the operating system is used by processes to duplicate themselves. The Call parent process may use this system to establish a child process, and the parent process's execution will be suspended while the child process runs.

exec(): This system call is called when an executable file replaces an earlier executable file inside the context of an active process. Even though a new process is not formed, the existing process identification remains; instead, the new process replaces the old one's stack, data, head, data, and so on.

kill(): The kill() system function is used by the operating system to notify processes to terminate. A kill system call has many implications and is not necessarily used to terminate a process.

exit(): An exit system call is used to terminate a program. The resources that the process was utilizing were freed when the exit system call was performed.

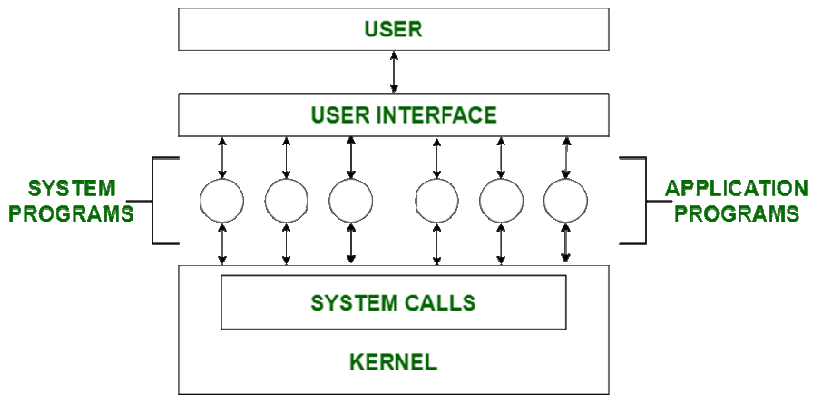


Figure 4.1: Important system calls in systems

File Management:

A file is a collection of specified information saved in a computer system's memory. File management is described as the process of managing files in a computer system, which includes the creation, modification, and deletion of files[5].

a. It is beneficial to create new files in the computer system and save them in certain areas.
b. It aids in swiftly and simply finding these files on the computer system.
c. It simplifies and user-friendly the process of sharing files among several users.
d. It is beneficial to keep files in distinct folders known as directories.
e. These folders allow users to rapidly search for files or organise them based on their intended usage.
f. It allows users to change the data of files or the names of files in directories.

Status Information: Some users request information such as the date, time, amount of available RAM, or disc space. Others provide more thorough performance, logging, and debugging information. All of this data is prepared and displayed or printed on output devices. The output of programmes is shown via a terminal or other output devices or files, or a GUI window.

File Modification: utilize this to change the contents of files. We utilised several sorts of editors for files saved on drives or other storage media. We utilise specific commands to search the contents of files and conduct file manipulations.

Programming-Language Support: We utilise Compilers, Assemblers, Debuggers, and Interpreters that are already available to users for popular programming languages. It offers complete user assistance. We are capable of running any programming language. All important languages have previously been given.

Program Loading and Execution - After Assembling and Compiling, the programme must be loaded into memory for execution. A loader is a component of an operating system that is in charge of loading applications and libraries. It is one of the most important steps in establishing a programme. The system includes loaders, composable loaders, connection editors, and Overlay loaders.

Communications: Programs establish virtual links between processes, people, and computer systems. Users may transmit messages to another user on their screen, send e-mail, browse web sites, log in remotely, and transfer files from one user to another[6].

Some examples of system program in O.S. are –

A. Windows 10
B. Mac OS X
C. Ubuntu
D. Linux
E. Unix
F. Android
G. Anti-virus
H. Disk formatting
I. Computer language translators

System programmes provide an environment in which programmes may be written and run. In the most basic sense, system programmes serve as a link between the user interface and system calls. In actuality, they are much more complicated. A compiler, for example, is a complicated system programme.

System Programs Function

The system programme is a component of the operating system. It is often located between the user interface and system calls. Because system calls are what people engage with and system programmes are closer to the user interface, system programmes establish the user perception of the system rather than system calls[7]. The following is a picture that depicts system programmes in the operating system hierarchy:

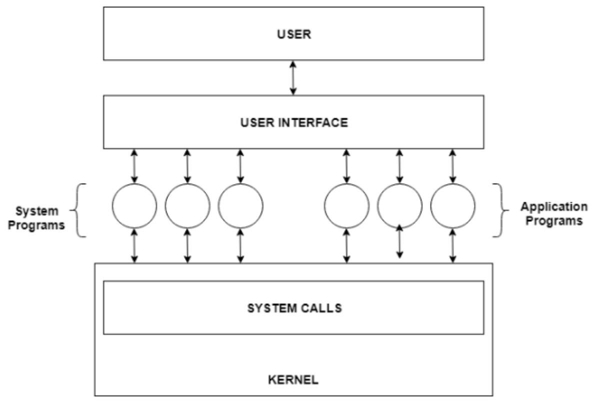


Figure 4.2: System Programs Function

Both system and application programmes serve as a link between the user interface and system calls. As a result, the operating system perceived by the user is the system programmes rather than the system calls.

System Program Types

System programmes are separated into seven sections. These are listed below:

Status Information: The state information system programmes give essential data about the system's present or previous status. This may comprise the system date, time, available RAM, storage space, logged in users, and so on.

Communications: These system applications, such as web browsers, are required for system communications. Web browsers enable computers to connect with one another and retrieve information from the network as needed.

File Manipulation: These system applications allow you to modify system files. This may be accomplished by the use of numerous commands such as create, delete, copy, rename, print, and so on. These commands can create and destroy files, copy the contents of one file to another, rename and print files, and so on.

Program Loading and Execution: System programmes responsible for programme loading and execution ensure that programmes can be loaded into memory and executed properly. Loaders and Linkers are two examples of such system programmes.

File Modification: File modification system applications update the contents in the file or modify it in some other manner. Text editors are an excellent example of file modification system software.

Application Programs: Application programmes may provide a variety of services based on the demands of the consumers. Database programmes, word processors, charting tools, spreadsheets, games, and scientific applications are examples.

Programming Language Support: These system programmes offer extra programming language support functionalities. Compilers, debuggers, and other similar tools are instances of this. These, in turn, build a programme and ensure that it is error-free[8].

The primary distinctions between System Call and System Program in the Operating System

In the operating system, there are many fundamental distinctions between System Call and System Program. The following are some key distinctions between System Call and System Program:

Using the system call, a user may request access to the operating system's services. The system programme, on the other hand, fulfils a frequent user requirement and offers a suitable environment for a programme to design and execute efficiently.

Using high-level languages such as C and C++, the programmer generates system calls. The assembly level language is used to generate calls that interface directly with the system's hardware. A programmer, on the other hand, only utilises a high-level language to develop system programmes.

System calls define the interaction between the OS's services and the user process. The system programme, on the other hand, specifies the user interface of the operating system.

The system programme fulfils the high-level request of the user application. It breaks the request down into many low-level requests. The system call, on the other hand, fulfils the user program's low-level demands.

A system call is used by the user process to request an OS service. The system programme, on the other hand, converts the user request into a collection of system calls required to fulfil the need.

System calls are classified as either file manipulation, device manipulation, communication, process management, information maintenance, or protection. A System programme, on the other hand, may be classified as file management, programme loading and execution, programming-language support, status reporting, communication, and file modification.

Starting up an Operating System:

When we push the power button on our computer, all of the gadgets receive electricity and are initialised. Because RAM is volatile memory, our primary memory, which is responsible for storing instructions, will be initially empty. As a result, a tiny set of instructions will be present in the non-volatile memory known as ROM. These instructions will be transmitted to the CPU, where they will be executed, checking all of the hardware attached to the system. If there are any issues with the hardware, we will be notified through beeps or on-screen notifications. After the hardware has been tested, the booting procedure proceeds to load the operating system. Non-volatile memory instructions are hardwired on the motherboard and cannot be deleted. BIOS, which stands for Basic Input Output System, refers to the tiny set of instructions included in the ROM[9].

Booting Type: There are two kinds of Booting available:

Cold Booting/Hard Booting: Cold booting refers to the procedure by which our computer system transitions from the shutdown state to the startup state by hitting the power button. The system reads the BIOS from ROM before loading the operating system.

Warm Booting/Soft Booting: Warm booting is the process of restarting the computer for reasons such as configuring freshly installed software or hardware. Warm booting is referred to as restarting.

Operating System Booting Procedure

When we turn on our computer, it might be started by hardware, such as a button push, or by software instruction. Because a computer's central processing unit (CPU) does not have software in its main memory, some process must load software into main memory before it can be run. The six stages to define the boot process in the operating system are as follows:

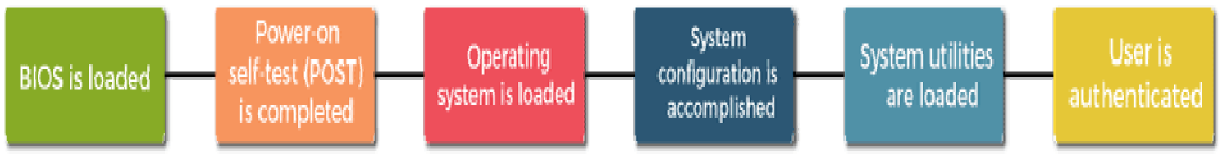

BIOS is loaded
Power-on self-test (POST) is completed
Operating system is loaded
System configuration is accomplished
System utilities are loaded
User is authenticated

Figure 4.3: Operating system booting procedure

Step 1: When the computer system is powered on, the BIOS (Basic Input/Output System) executes a series of activities or functionality tests on programmes stored in ROM, known as POST (Power-on Self Test), to determine whether or not the system's peripherals are in perfect working order.

Step 2: Once the BIOS has completed the pre-boot activities or capability tests, it reads the bootable sequence from the CMOS (Common Metal Oxide Semiconductor) and searches for the master boot record in the first physical sector of the bootable disc as stated in the CMOS boot device sequence. For instance, suppose the boot device sequence is:

Floppy Disk

Hard Disk

CDROM

Step 3: After that, the master boot record will look for a floppy disc drive first. If the master boot record is not discovered, the hard disc drive will look for it. The CDROM drive will seek if the master boot record is not even available on the hard disc. If the system is unable to read the master boot record from any of these sources, the ROM displays "No Boot device identified" and the system is shut down. When a bootable disc drive's master boot record is found, the operating system loader, also known as the Bootstrap loader, is loaded into memory from the boot sector of that bootable drive. A bootstrap loader is a specific application found in a bootable drive's boot sector.

Step 4: The bootstrap loader imports the IO.SYS file first. Following that, the MSDOS.SYS file, which is the core file of the DOS operating system, is loaded.

Step 5: After that, the MSDOS.SYS file looks for the Command Interpreter in the CONFIG.SYS file and loads it into memory. If no Command Interpreter is provided in the CONFIG.SYS file, the COMMAND.COM file is loaded as the DOS operating system's default Command Interpreter.

Step 6: The last file to load and run is the AUTOEXEC.BAT file, which includes a series of DOS instructions. Following that, the prompt appears. The drive letter of the bootable drive is shown on the computer system, indicating that the operating system was successfully installed on the machine from that drive.

System Boot

1. When we hit the power button, all of the system's components are powered up and initialised. After the CPU has been initialised, instructions must be executed.
2. A short collection of instructions known as BIOS is loaded from ROM.
3. After completing the Power-On-Self-Test (POST), the BIOS will locate the bootable sequence in the CMOS.
4. It detects the first bootable device based on the bootable sequence.
5. The first bootable device loads instructions from the Master Boot Record, which is located in logical Sector 0.
6. This collection of instructions includes information about the Boot Loader, which is responsible for loading the operating system.
7. This boot loader information is individual to each operating system; for example, the boot loader for Linux is GRUB (GRand Unified Bootloader)[10].

After that, the boot loader loads the operating system into memory.Finally, all of the critical system files and drivers are loaded into memory, and control is given to the operating system.

REFERENCES

[1] D. Simion, A. Purcărea, A. Cotorcea, and F. Nicolae, “Maintenance onboard ships using computer maintenance management system,” Sci. Bull. Nav. Acad., 2020, doi: 10.21279/1454-864X-20-I1-017.

[2] L. Volontyr and O. Zelinska, “COMPUTER SYSTEMS DESIGN MANAGEMENT,” Eng. ENERGY, Transp. AIC, 2019, doi: 10.37128/2520-6168-2019-3-15.

[3] C. Chen and Z. Hui, “Computer Network System Security Management and Maintenance Strategy,” in Journal of Physics: Conference Series, 2020. doi: 10.1088/1742-6596/1533/2/022057.

[4] J. Liu and K. Li, “Design and implementation of computer aided equipment management information system,” Comput. Aided. Des. Appl., 2020, doi: 10.14733/CADAPS.2021.S1.155-164.

[5] G. Wu, “Computer Finance Management System Innovation Thinking,” in Journal of Physics: Conference Series, 2020. doi: 10.1088/1742-6596/1486/5/052025.

[6] B. Sidawi and A. A. Al-Sudairi, “The use of advanced computer based management systems by large Saudi companies for managing remote construction projects,” in Procedia Engineering, 2014. doi: 10.1016/j.proeng.2014.07.013.

[7] W. Zhang, C. Zhang, C. Han, and H. Li, "Development Trend Analysis of Computer Management Information System," J. Electron. Res. Appl., 2020, doi: 10.26689/jera.v4i1.1146.

[8] N. Yalcin, Y. Altun, and U. Kose, "Educational material development model for teaching computer network and system management," Comput. Appl. Eng. Educ., 2015, doi: 10.1002/cae.21636.

[9] Z. Lin, "Design and Implementation of WEB-based Computer Experiment Management System," Int. J. Web Appl., 2017.

[10] W. Velasquez and M. Filian-Gomez, "Communication Network Model for a Computer Management and Control System implemented using FIWARE platform: Case Study," IEEE Lat. Am. Trans., 2020, doi: 10.1109/TLA.2020.9400434.

CHAPTER 5

PROCESS MANAGEMENT

Dr. Yashaswini K A, Assistant Professor
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- yashaswini@presidencyuniversity.in

Process in an operating system

A process is simply software that is in use. Any procedure must be carried out in a precise sequence. A process is an object that aids in the representation of the basic unit of work that must be implemented in any system. In other words, we develop computer programmes in the form of text files, and when we execute them, they transform into processes that do all of the tasks described in the programme. When a programme is loaded into memory, it may be divided into four parts: stack, heap, text, and data. The following diagram is a simplified illustration of a process in main memory[1].

A Process's Components

It is broken into four portions as follows:

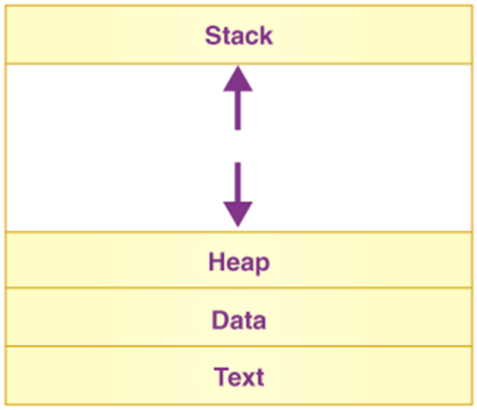


Figure 5.1: Process components

Stack

The process stack stores temporary data such as method or function arguments, return address, and local variables.

Heap

This is the memory that a process dynamically allocates while it is running.

Text This includes the contents of the processor's registers as well as the current activity as represented by the programme counter value.

Data

This section includes both global and static variables.

Life Cycle of a Process

A process passes through numerous phases while it runs. Distinct operating systems have different phases, and the nomenclature of these states vary. A process may, in general, be in one of the five stages described below at any given moment.

Start

This is the state in which a process is in when it is initially started/created.

Ready

The process is currently waiting for a processor to be allocated to it. Processes that are ready are waiting for the operating system to allocate them a processor so that they may execute. The process may reach this state after beginning or while running, however it may be interrupted by the scheduler to give the CPU to another process.

Running

The process state is set to running when the OS scheduler allocates a processor to it, and the processor executes the process instructions.

Waiting

When a process has to wait for a resource, such as user input or the availability of a file, it enters the waiting state.

Exit or Terminated

Once the process has finished its execution or has been terminated by the operating system, it is moved to the terminated state, where it awaits removal from main memory[2].

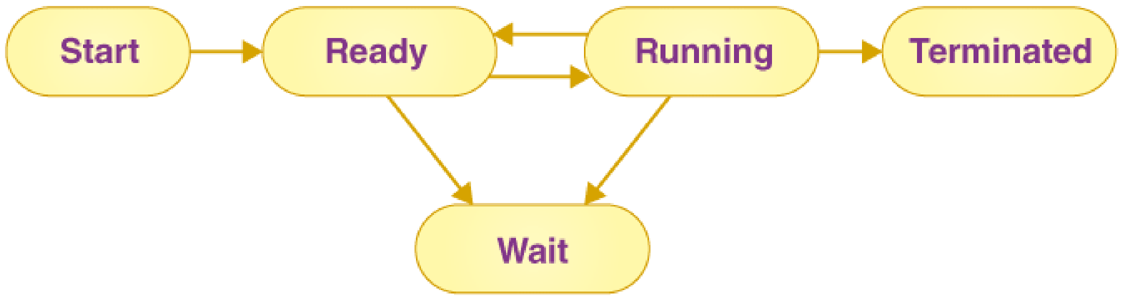


Figure 5.2: Batch operating system

Process Control Unit (PCB)

A process control block, which is a data structure maintained by the operating system, is present in every process. The PCB is identified using an integer process ID (or PID). As seen here, PCB holds all of the information needed to keep track of a process.

The current condition of the process

The current condition of the process, such as whether it is ready, waiting, running, or whatever.

Privileges in the process

This is essential for granting or denying access to system resources.

Process Identification

Each process in the operating system has its own unique identity.

Pointer

It is a pointer pointing to the parent process.

The programme counter is a pointer that leads to the address of the process's next instruction.

Registers on the CPU

In the running state, processes must be stored in multiple CPU registers.

CPU scheduling details

For the process to be scheduled, the process priority and further scheduling information are necessary.

Data on memory management

This contains data from the page table, memory restrictions, and segment table, all of which are affected by the amount of memory consumed by the operating system.

Accounting data includes, among other things, CPU use for process execution, time limits, and execution ID.

Information regarding IO status

A list of the process's I/O devices is included in this section.

The PCB architecture is entirely reliant on the operating system, and various operating systems may include varying amounts of information. Below is a simplified schematic of a PCB[3], [4].

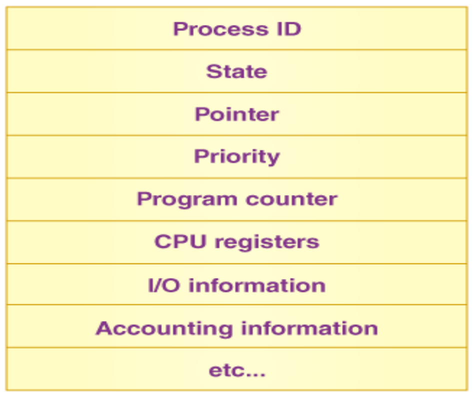


Figure 5.3: Process control unit

The PCB is retained throughout a procedure and then removed after the operation is completed.

The Various Process States

The processes of the operating system may be in one of the following states:

1. NEW - The process's development.
2. READY - Awaiting the process to be allocated to any processor.
3. RUNNING - The execution of instructions.
4. WAITING - The process's anticipation of an upcoming event (such as I/O completion or signal receiving).
5. FINISHED - A process has finished its execution.

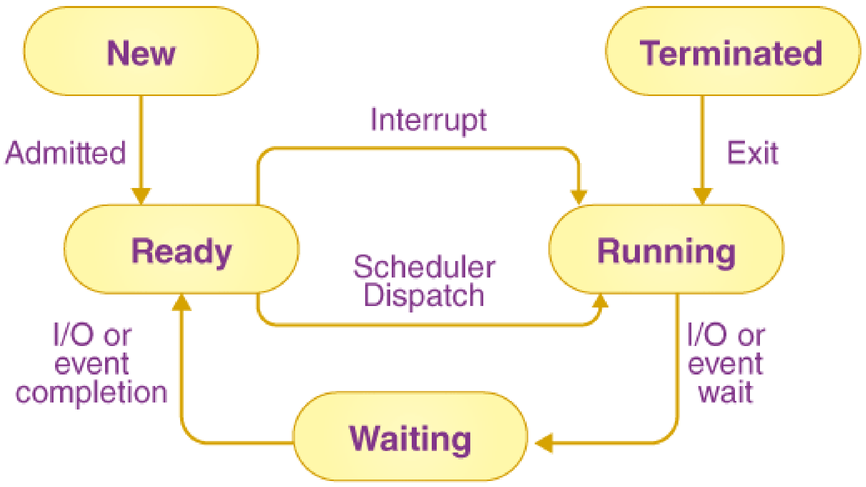


Figure 5.4: Processes of the operating system

Program vs. Process

A programme is a piece of code that may range from a single line to millions of lines. A programmer generally creates a computer programme in a programming language. The process, on the other hand, is effectively a representation of the currently executing computer programme. It has a rather limited lifespan.

A computer programme is a collection of instructions that, when executed by a computer, accomplish a certain task. When we compare a programme to a process, we may determine that a process refers to a dynamic instance of a computer programme. An algorithm is a component of a computer programme that performs a certain function. A software package is a set of computer applications, libraries, and data[5], [6].

Process Planning

When there are many processes that may be executed, the operating system decides which one to run first; this is known as process scheduling.

A scheduler is a software that makes decisions based on a scheduling algorithm. A good scheduling algorithm has the following characteristics:

1. Response time for users should be maintained to a bare minimum.
2. The total number of tasks handled each hour should be as large as feasible, suggesting that an effective scheduling system should deliver the most throughput possible.
3. The CPU's full capacity should be used.
4. Equal CPU time should be given to each process.
5. Process Conceptualization When considering operating systems, one dilemma that comes up is what to label all of the CPU activity.

A batch system runs jobs, while a time-shared system runs user applications, sometimes known as tasks. Even on a single-user system like Microsoft Windows, a user may be able to run many applications at the same time, such as a word processor, a web browser, and an e-mail client. Even though the user can only run one application at a time, the operating system may need assistance for internal programmed operations like memory management. Because all of these tasks are comparable in many ways, we refer to them as processes. In this article, the words task and process are practically interchangeable. Although we prefer the word process, much of operating-system theory and nomenclature was established when task processing was the primary activity of operating systems. It would be deceptive to discontinue using popularly recognised words including the word job (such as work scheduling) just because process has surpassed job[7].

The Procedure

As previously stated, a process is a programme in execution. A process is more than just the computer code, often known as the text portion. It also comprises the processor's current activity, as indicated by the programme counter value and the contents of the processor's registers. In addition to the process stack, which holds temporary data (such as function arguments, return addresses, and local variables), a process also comprises a data section, which contains global variables.

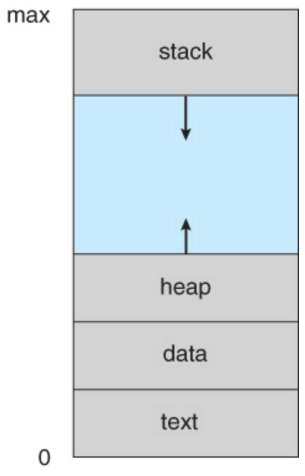


Figure 5.5: Processes of the operating system

A process may also have a heap, which is memory that is dynamically allocated throughout the process's execution. We emphasise that a programme is not a process in and of itself; a programme is a passive entity, such as a file containing a list of instructions stored on disc (commonly referred to as an executable file), whereas a process is an active entity, with a programme counter indicating the next instruction to execute and a set of associated resources. When an executable file is put into memory, it becomes a process. Double-clicking an executable file icon or putting the executable file's name on the command line are two typical methods for loading executable files (as in prog. exe or a. out.) Even though two processes are connected with the same programme, they are treated as two distinct execution sequences. For example, many users may be running distinct copies of the mail programme, or the same user may be running multiple instances of the web browser software. Each of them is a distinct process, and although the text parts are identical, the data, heap, and stack sections differ. It is also usual for a process to generate many processes while it runs.

1. Process State
2. A process's state changes as it runs. The present activity of a process defines the state of that process in part. Each process might be in one of three states:
3. New. The procedure is being developed.
4. Exercise. Instructions are being carried out.
5. I'm waiting. The procedure is waiting for an event to occur (such as I/O completion or signal receiving).
6. Ready. The procedure is currently awaiting assignment to a processor.
7. Discontinued.

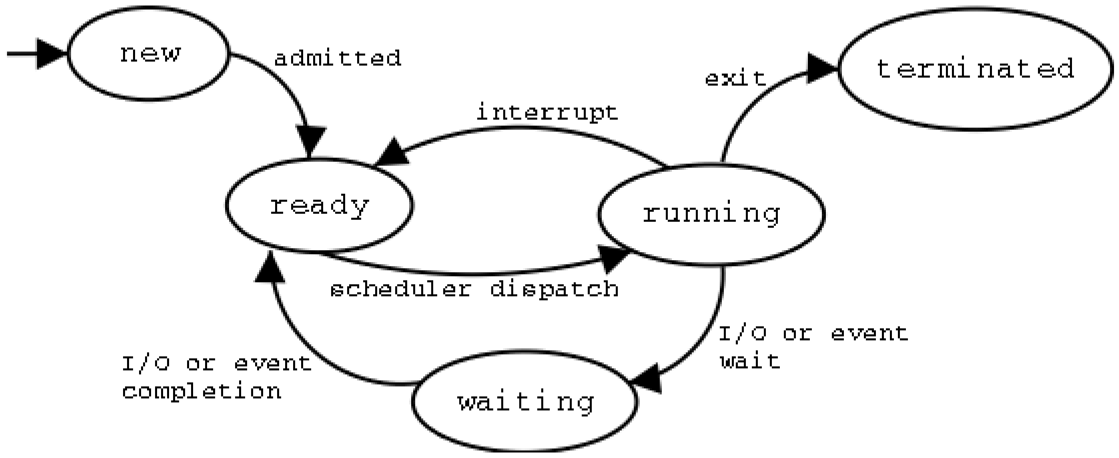


Figure 5.6: Processes of the operating system

The procedure has been completed. These names are arbitrary and vary across operating systems. However, the states they represent are fotind on all platforms. Certain operating systems also designate process states more precisely. It is critical to understand that only one process may execute on any given CPU at any one time. Many procedures, however, may be ready but limited.

Process Control Block (PCB)

A process control block (PCB)—also known as a task control block—represents each process in the operating system. It comprises several bits of information related to a certain process, including the following:

The condition of the process. The status might be fresh, ready, operating, waiting, or stalled, for example[8].

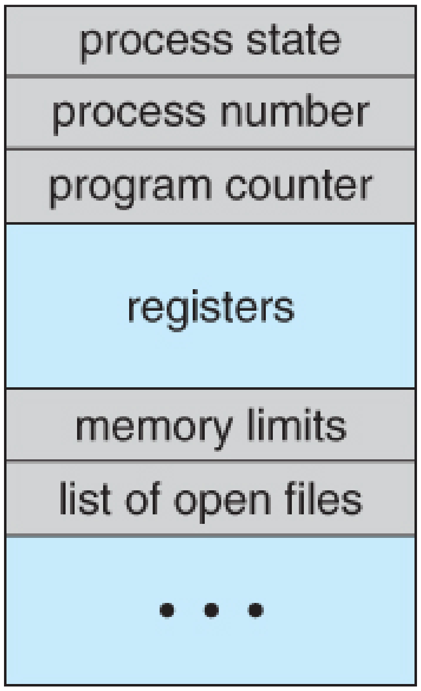


Figure 5.6: Processes Control Block

A counter for the programme. The address of the next instruction to be executed for this process is indicated by the counter.

Registers on the CPU. The number and kind of registers varies depending on the computer architecture. Accumulators, index registers, stack pointers, and general-purpose registers, as well as any condition-code information, are included. This state information, along with the programme counter, must be preserved when an interrupt occurs in order for the process to continue appropriately thereafter.

• CPU scheduling data. This data provides a process priority, scheduling queue pointers, and any other scheduling settings.

Memory-management knowledge. Depending on the memory system used by the operating system, this information may contain the value of the base and limit registers, the page tables, or the segment tables.

Accountancy data. This data comprises CPU and real-time use, time limitations, account numbers, job or process numbers, and so on.

Status of I/O devices. This data comprises a list of I/O devices assigned to the process, as well as a list of open files. In summary, the PCB merely acts as a storage for any information that may differ from one procedure to the next[9].

Threads

The process model outlined so far has assumed that a process is a programme that executes on a single thread. When a process runs a word-processor programme, for example, a single thread of instructions is performed. Because of this one thread of control, the process can only accomplish one job at a time.

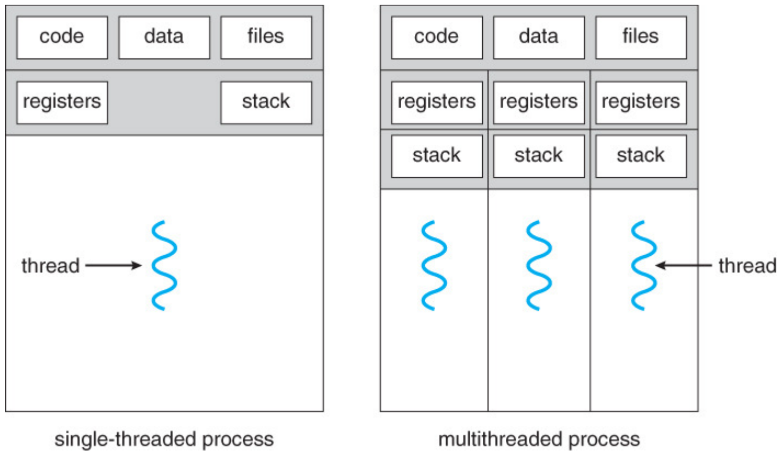


Figure 5.7: CPU scheduling data

For example, the user cannot type in characters and utilise the spell checker at the same time. Many current operating systems have expanded the idea of a process to enable it to have many threads of execution and so accomplish more than one job at the same time.

Processes operations: A process is a programme in execution that is more than just a piece of computer code known as a text section. This notion applies to all operating systems since all tasks performed by the operating system need the usage of a process.

When the process's status changes, it executes. The present activity of a process defines the process's state.

Each procedure may be in any of the states listed below. –

1. A new method is being developed.
2. Running In this condition, the instructions are being carried out.
3. Waiting the process is in a waiting state until an event happens, such as I/O operation completion or signal reception.
4. Prepared the procedure is ready to be allocated to a processor.
5. Completed the procedure has been completed.

It is critical to understand that only one process may execute on any given CPU at any one time. Many procedures may be prepared and ready to go.

Process Operations

Process's two primary operations are as follows:

Process Development

There should be four major events that cause processes to emerge.

Initialization of the system

When an operating system is booted, several processes are launched. Some of them are as follows:

Foreground processes are those that engage with people and do their task for them.

Background processes are also known as daemons and are not connected with individual users, but rather perform a specific purpose.

A running process executes a process-creation system call.

To assist it in doing its work, the running process will send system calls to generate one or more new processes. A user request to launch a new process

A new process is formed by performing a process creation system call on an existing process.

The system call used to start a new process in UNIX is fork ()

CreateProcess() in Windows provides ten arguments for handling both process creation and loading the right software into the new process.

The start of a batch task

Users will submit batch tasks to the system.

When the operating system starts a new process and executes the next task in the input queue.

Termination of the Process

A call to kill in UNIX or End Process in Windows will terminate the process.

The process has been ended for the following reasons:

a. Normal exit Most processes end when their task is complete and issue a system call to exit.
b. Error exit The third kind of error results from software defects such as performing an invalid instruction, referring, or dividing by zero.
c. Fatal exit A process terminates when it detects a fatal error.
d. Killed by another process A process makes a system call to terminate another process.

Various Operations on Processes

Processes may be subjected to a variety of procedures. Process creation, process preemption, process blocking, and process termination are a few examples. These are listed in detail below.

Process Development

For various activities, processes must be formed in the system. This may be accomplished by the following events:

a. Process creation request from a user
b. System initialization
c. A running process executing a process creation system call
d. Initialization of batch jobs

Another process may use fork to establish a process (). The producing process is referred to as the parent process, while the generated process is referred to as the child process. A child process can only have one parent, but a parent process may have several children. The memory image, open files, and environment strings are shared by the parent and child processes. They do, however, have different address spaces[10].

The figure below shows how to create a process using fork().

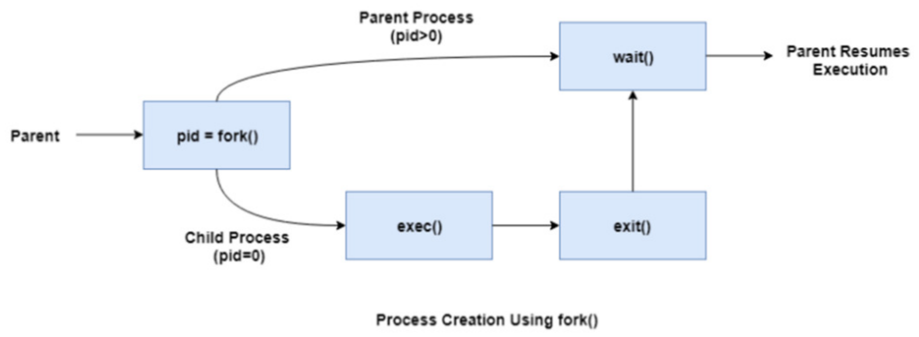


Figure 5.8: Process Creation Using fork()

Preemption of the Process

In preemption, an interrupt mechanism is used to suspend the current process, and the next process to run is decided by the short-term scheduler. Preemption ensures that all processes have access to CPU time for execution. The following figure illustrates process preemption:

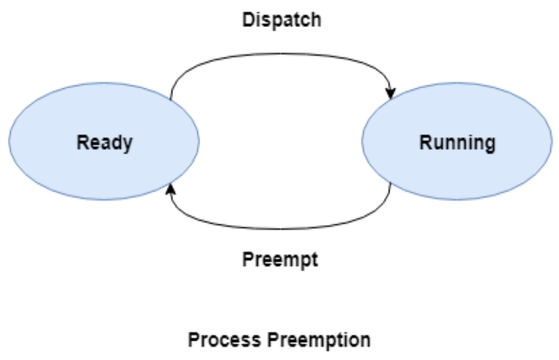


Figure 5.9: Process Preemption

Process Obstruction

If the process is waiting for anything to happen, it is blocked. This event might be I/O since I/O events run in main memory and do not need the CPU. When the event is finished, the process returns to the ready state. The following diagram depicts process blockage.

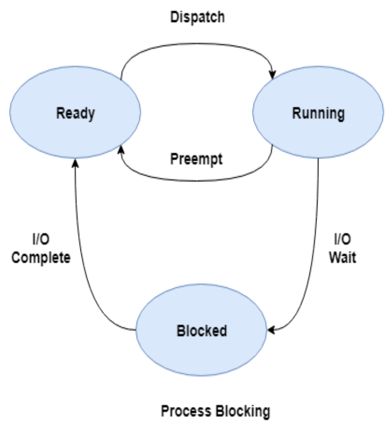


Figure 5.10: Process Obstruction

Termination of the Process

The process is terminated after it has finished the execution of its final instruction. After a process is terminated, its resources are relinquished. If a child process's job is no longer important, its parent process may terminate it. Before terminating, the child process communicates its status information to the parent process. Furthermore, when a parent process is ended, its child processes are also terminated, and the child processes cannot function if the parent processes are terminated.

REFERENCES

[1] A. De Ramón Fernández, D. Ruiz Fernández, and Y. Sabuco García, "Business Process Management for optimizing clinical processes: A systematic literature review," Health Informatics J., 2020, doi: 10.1177/1460458219877092.

[2] J. Mendling, B. T. Pentland, and J. Recker, "Building a complementary agenda for business process management and digital innovation," European Journal of Information Systems. 2020. doi: 10.1080/0960085X.2020.1755207.

[3] J. Mendling et al., "Blockchains for business process management - Challenges and opportunities," ACM Trans. Manag. Inf. Syst., 2018, doi: 10.1145/3183367.

[4] M. Fischer, F. Imgrund, C. Janiesch, and A. Winkelmann, "Strategy archetypes for digital transformation: Defining meta objectives using business process management," Inf. Manag., 2020, doi: 10.1016/j.im.2019.103262.

[5] J. A. Garcia-Garcia, N. Sanchez-Gomez, D. Lizcano, M. J. Escalona, and T. Wojdynski, "Using Blockchain to Improve Collaborative Business Process Management: Systematic Literature Review," IEEE Access, 2020, doi: 10.1109/ACCESS.2020.3013911.

[6] E. Lamine, R. Thabet, A. Sienou, D. Bork, F. Fontanili, and H. Pingaud, "BPRIM: An integrated framework for business process management and risk management," Comput. Ind., 2020, doi: 10.1016/j.compind.2020.103199.

[7] A. Bitkowska, "The relationship between Business Process Management and Knowledge Management - selected aspects from a study of companies in Poland," J. Entrep. Manag. Innov., 2020, doi: 10.7341/20201616.

[8] M. Röglinger, J. Pöppelbuß, and J. Becker, "Maturity models in business process management," Business Process Management Journal. 2012. doi: 10.1108/14637151211225225.

[9] S. Mishra, K. K. Sree Devi, and M. K. Badri Narayanan, "People & process dimensions of automation in business process management industry," Int. J. Eng. Adv. Technol., 2019, doi: 10.35940/ijeat.F8555.088619.

[10] I. Usman, N. H. Hartani, and M. Sroka, "Operational performance of sme: The impact of entrepreneurial leadership, good governance and business process management," Polish J. Manag. Stud., 2020, doi: 10.17512/pjms.2020.21.1.30.

CHAPTER 6

INTER PROCESS COMMUNICATION (IPC)

Dr C Kalaiarasan, Professor & Asso Dean
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- kalairasan@presidencyuniversity.in

One of the major technologies employed by operating systems to fulfil these aims is interprocess communication (IPC). IPC enables processes to interact with one another without the need for user-level functions or interfaces. It enables separate components of a software to access shared data and files without interfering with one another. Messages are sent between two or more processes in inter-process communication. Processes might run on the same or distinct machines. In this essay, we will explore IPC and its importance, as well as many methods to IPC. Interprocess communication (IPC) is a method of sharing information between processes in a computer system. IPC allows many applications to execute in parallel, exchange data, and interact with one another. It is significant for two reasons: First, it accelerates task execution; second, it assures that tasks are completed properly and in the sequence in which they were initiated[1].

Process 1
Process 2

Interprocess communication is required.

IPC allows many applications to execute in parallel, exchange data, and interact with one another. It is significant for two reasons:

A. It expedites work completion.
B. It guarantees that the tasks are completed properly and in the right sequence.
C. IPC is required for an operating system to function properly.
D. IPC is used by operating systems to exchange data with tools and components used by the system to communicate with the user, such as the keyboard, mouse, and graphical user interface (GUI).

IPC also enables the system to execute numerous applications concurrently. For instance, the system may utilise IPC to tell the windowing system about the state of a window on the screen.

In comparison to a real-world example

When utilising a team-based system, you must interact with all of the company's teams. The Business Objects team, the product sales team, and the technical teams are among them. Communication between them will speed up the job and boost overall efficiency. Similarly, an operating system enables a certain software to handle several user requests at the same time. Overall efficiency will improve[2].

The Benefits of Interprocess Communication

A. Interprocess communication allows one programme to manage another and provides for error-free data exchange.
B. Interprocess communication aids in the efficient transmission of messages between processes.
C. Because the software is broken into discrete portions of code that function independently, it is simple to maintain and debug.
D. Programmers may do a number of other things at the same time, such as editing, listening to music, compiling, and so on.
E. Data may be transferred between many apps at the same time.
F. Tasks may be partitioned and executed on specialised processors. The data may then be exchanged via IPC[3].

Interprocess communication disadvantages

A. The application is unable to write to comparable places.
B. Processes or applications that utilise the shared memory paradigm must avoid writing to comparable memory regions.
C. The shared storage approach might pose issues such as storage synchronisation and protection, which must be handled; and
D. It is slower than a direct function call.

Role of IPC in computer system

When an application delivers a message to an operating system process, IPC happens. The message is sent by the operating system to a selected IPC mechanism, which processes it and returns a response to the application. IPC techniques may be located in an operating system's kernel or user space.

IPC is a critical process in computer system functioning. It allows many applications to operate in parallel, exchange data, and interact with one another. IPC is critical for an operating system's efficiency because it guarantees that activities are completed appropriately and in the sequence in which they were initiated.

Implementation of Inter process Communication

Message transmission

Message forwarding is another essential method of inter-process communication with other processes. When two or more processes communicate with one another, each sends messages to the others through the Kernel. Here's an example of two processes exchanging messages: – In this case, the process sends an "M" message to the OS kernel. Process B then reads this message. A communication connection between the two processes is essential for message exchange to be effective. There are numerous methods for establishing these connections[4].

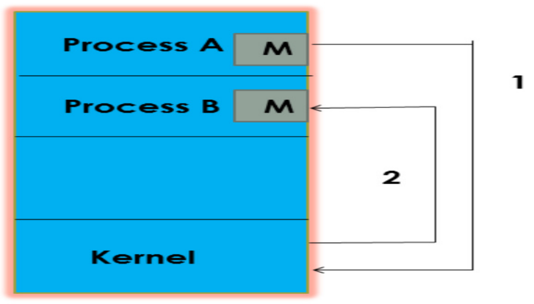


Figure 6.1: Message transmission

Shared memory

Shared memory is memory that is shared by all processes that are formed utilising shared memory. By coordinating access amongst all processes, this form of memory should defend itself. Both processes, such as A and B, may create a shared memory segment and exchange data across it. Shared memory is crucial for the following reasons:

a. It is a technique of transmitting data across processes;
b. It is considerably quicker and more reliable than these approaches. Shared memory enables two or more processes to access the same data copy.

Assume Process a wishes to connect with Process B and requires its address space to be attached to this shared memory segment. Process A will send a message to the shared memory, and Process B will read that message. As a result, processes are responsible for guaranteeing synchronisation such that neither process writes to the same place at the same time[5].

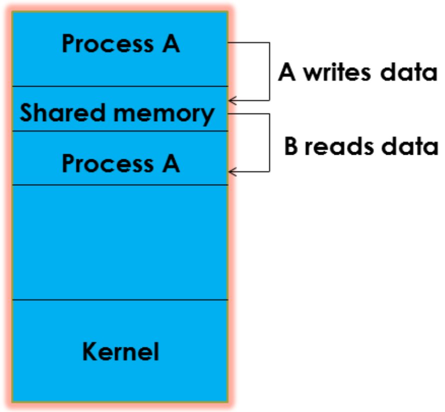


Figure 6.2: Shared Memory

3. Pipes

Pipes are a form of data conduit that is typically used to communicate one-way between two programmes. Because this is a half-duplex approach, the main and secondary processes communicate. Additional lines, however, are necessary to reach complete duplex. Between the two operations, the two pipes form a bidirectional data channel. However, just one pipe generates a unidirectional data channel. Pipes are most often seen in Windows operating systems. As indicated in the figure, one process will transmit a message to the pipe. The message will be obtained and sent to standard output by another process.

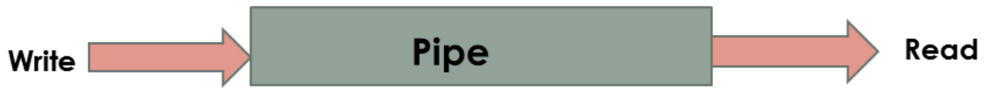


Figure 6.3: Pipe transmission

5. **Signal**

The signal is a communication facility that enables processes to communicate with one another. A signal informs a process that it needs to perform something. A signal may be sent from one process to another. A signal also enables one activity to interrupt another. A signal is a mechanism for processes to communicate with one another.

CPU allocation

CPU scheduling is a technique that enables one process to use the CPU while another process's execution is on hold (in a waiting state) due to the unavailability of any resource such as I/O, etc., allowing the CPU to be fully used. The goal of CPU scheduling is to make the system as efficient, quick, and fair as possible.

The operating system must choose one of the processes in the ready queue to run whenever the CPU becomes idle. The short-term scheduler is in charge of the selecting process (or CPU scheduler). The scheduler chooses one of the processes in memory that is ready to run and assigns the CPU to it.

Processor Scheduling: Dispatcher

The Dispatcher is another component engaged in the CPU scheduling process. The dispatcher is the module that assigns CPU control to the process chosen by the short-term scheduler. This procedure entails the following steps:

• Switching context

• Switching to user mode

• Navigating to the correct position in the user software to restart it from where it was last left.

Given that it is run at every process changeover, the dispatcher should be as quick as feasible. Dispatch Latency is the time it takes the dispatcher to halt one operation and start another. The following diagram explains dispatch latency[5]:

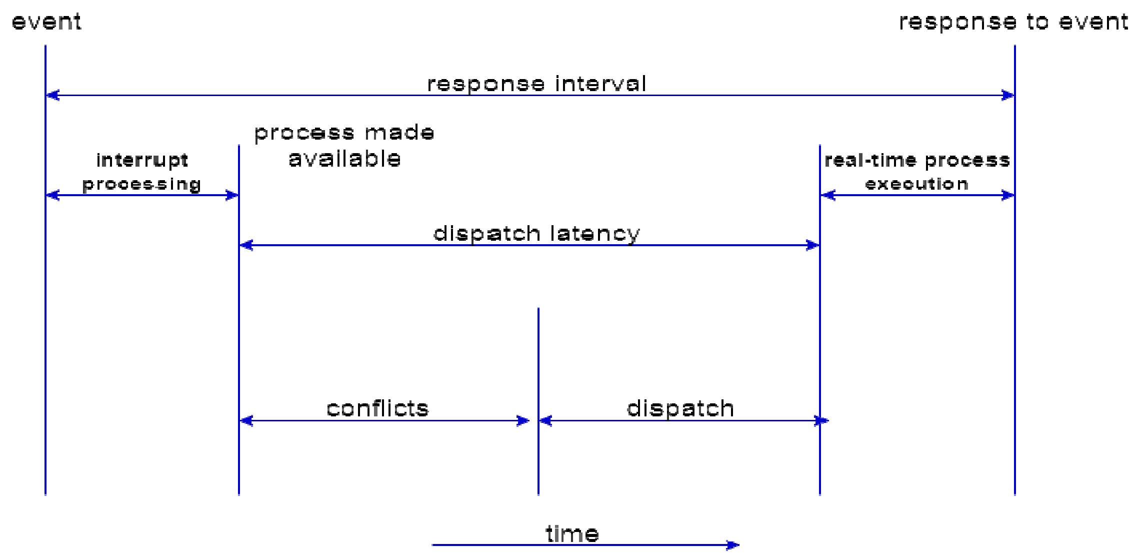


Figure 6.4: CPU scheduling process

CPU Scheduling Varieties

CPU scheduling choices may be made under the following four conditions:

1. When a process transitions from the running to the waiting state (for an I/O request or the invocation of wait for one of the child processes to terminate).

2. When a process transitions from the running to the ready state (for example, when an interrupt occurs).

3. When a process transitions from the waiting to the ready state (for example, when I/O is completed).

4. When a procedure comes to an end.

There is no option in terms of scheduling in conditions 1 and 4. A new process must be chosen for execution (if one exists in the ready queue). However, there is an option in situations 2 and 3.

When scheduling occurs only in scenarios 1 and 4, we say the scheme is non-preemptive; otherwise, the system is preemptive.

Scheduling that is not preemptive

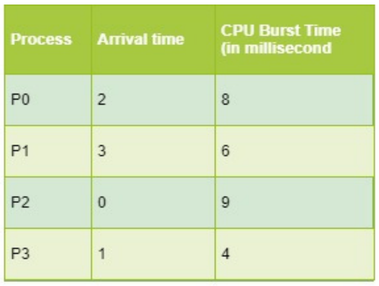
When the CPU is allotted to a process under non-preemptive scheduling, the process holds the CPU until it releases it, either by terminating or transitioning to the waiting state.

The Microsoft Windows 3.1 and Apple Macintosh operating systems also employ this scheduling approach.

Because it does not need the additional hardware (such as a timer) required for preemptive scheduling, it is the sole approach that can be utilised on certain hardware platforms.

Non-preemptive scheduling does not interrupt a running CPU process in the midst of its execution. Instead, it waits until the process's CPU burst period is complete before allocating the CPU to another process[6].

Some non-preemptive scheduling algorithms include: Shortest Job First (SJF) Scheduling and Priority (non-preemptive version) Scheduling, among others.



Process	Arrival time	CPU Burst Time (in millisecond
P0	2	8
P1	3	6
P2	0	9
P3	1	4

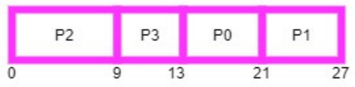


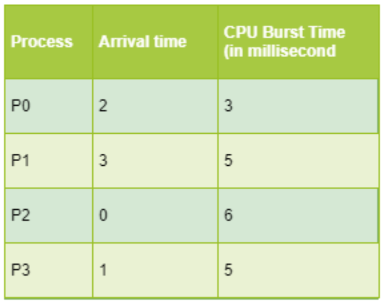
Figure 6.5: Non-Preemptive Scheduling

Preemptive Scheduling

Tasks are frequently allocated with priority in this sort of Scheduling. Although another activity is ongoing, it is sometimes required to perform a higher priority task first. As a result, the current job is halted for a short period of time before being restarted after the priority task has completed its execution.

Thus, this sort of scheduling is mostly employed when a process transitions from a running state to a ready state or from a waiting state to a ready state. The resources (that is, CPU cycles) are primarily provided to the process for a certain length of time before being removed, and the process is then put back in the ready queue if it still has CPU burst time left. That process remains in the ready queue until it is given the opportunity to run again.

Round Robin Scheduling (RR), Shortest Remaining Time First (SRTF), Priority (preemptive version) Scheduling, and others are preemptive scheduling algorithms[7].



Process	Arrival time	CPU Burst Time (in millisecond
P0	2	3
P1	3	5
P2	0	6
P3	1	5

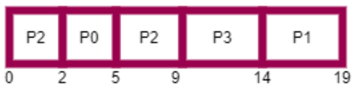


Figure 6.6: Preemptive Scheduling

CPU Scheduling: Criteria for Scheduling

When looking for the "best" scheduling method, consider the following criteria:

Utilization of the CPU

To make the maximum use of the CPU and avoid wasting any CPU cycles, the CPU should be active the majority of the time (ideally 100% of the time). In a practical machine, CPU use should vary between 40% (lightly loaded) to 90%. (Heavily loaded)

Throughput

It is the total number of processes accomplished per unit of time, or the total quantity of work completed in a unit of time. Depending on the method, this may vary from 10/second to 1 hour.

Turnaround Time is the amount of time it takes to execute a certain process, i.e. the time between when the process is submitted and when it is completed (Wall clock time).

Waiting Period

The total of the periods spent waiting in the ready queue is the length of time a process has been waiting in the ready queue to get CPU control.

Average Load

It is the average number of processes waiting in the ready queue to be processed by the CPU.

Response Time The moment it takes from the time a request is sent to the time the first response is delivered. Remember that the time until the first answer is important, not the conclusion of the process execution (final response).

For optimal optimization, CPU usage and throughput are often increased while other aspects are lowered[8], [9].

Operating System Scheduling Algorithms

Scheduling algorithms efficiently and effectively schedule tasks on the processor. A Process Scheduler is in charge of scheduling. It increases throughput to enhance CPU use.

The following are some prominent process scheduling methods that we will discuss in this chapter:

CPU Scheduling Algorithm Types

Process scheduling methods are classified into six kinds.

1. It is first come, first served (FCFS)

2. SJF (Shortest-Job-First) Scheduling

3. Minimal Remaining Time

4. Scheduling Priorities

5. Multilevel Round Robin Scheduling

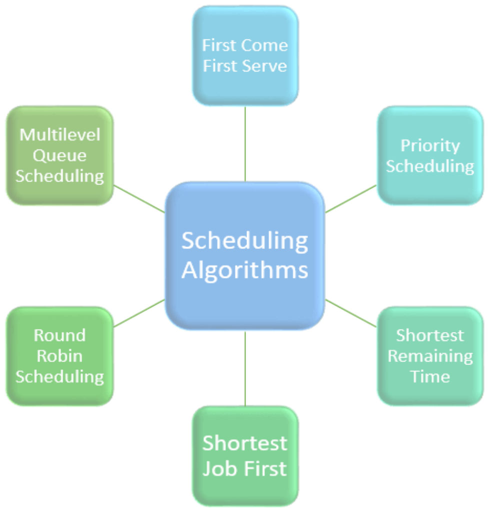


Figure 6.7: Operating System Scheduling Algorithms

First Come First Serve

FCFS is an abbreviation for First Come, First Serve. It is the most basic and straightforward CPU scheduling technique. The process that asks the CPU receives the CPU allocation first in this kind of method. A FIFO queue may be used to handle this scheduling strategy.

When a process joins the ready queue, its PCB (Process Control Block) is connected to the queue's tail. As a result, when a CPU becomes available, it should be allocated to the process at the top of the queue.

The FCFS technique has the following characteristics:

• It provides non-preemptive and pre-emptive scheduling algorithms.

• All jobs are completed on a first-come, first-served basis.

• It is simple to implement and utilise.

However, this approach performs poorly, and the whole wait time is extremely long.

hortest Remaining Time

SRT is an abbreviation for Shortest remaining time. It is often referred to as SJF preemptive scheduling. The process will be assigned to the task that is closest to completion in this way. This strategy avoids a newer ready state process from preventing an older process from completing.

SRT scheduling technique characteristics:

A. This approach is typically used in batch scenarios where brief tasks must be prioritised.
B. This is not the best way to do it on a shared system if the amount of CPU time needed is uncertain.

C. Assign the duration of the next CPU burst to each process. As a result, the operating system makes use of these lengths, which aids in scheduling the procedure in the smallest amount of time.

Scheduling Based on Priority

Priority scheduling is a way of arranging processes according to their importance. The scheduler chooses the tasks to work on based on their priority in this technique.

Priority Based Scheduling

Priority scheduling also aids OS in incorporating priority assignments. Processes with greater priority should be completed first, whereas tasks with equal priorities should be completed in a round-robin or FCFS fashion. Priority may be determined depending on memory limitations, time constraints, and so forth.

Scheduling in Rounds

Round robin is the most basic and oldest scheduling strategy. This method takes its name from the round-robin principle, which states that each individual receives an equal amount of something in turn. It is typically used for multitasking scheduling techniques. This algorithm approach aids in the execution of processes without regard for hunger.

Round-Robin Scheduling

• Round robin is a clock-driven hybrid approach.

• The time slice allotted to a certain job should be as short as possible. However, it may alter depending on the method.

• It is a real-time system that reacts to an event within a set time frame.

First, do the shortest job.

SJF (Lowest job first) is a scheduling method in which the process with the shortest execution time is chosen for execution next. This style of scheduling might be preemptive or non-preemptive. It decreases the average waiting time for other processes that are awaiting execution substantially.

SJF Scheduling Characteristics

• It is related with each work as a unit of completion time.

• When the CPU becomes available, the following process or task with the lowest completion time is run first.

• It is implemented in accordance with a non-preemptive policy.

• This algorithm technique is suitable for batch-type processing when waiting for tasks to finish is not a concern.

• It increases work production by providing shorter tasks that should be completed first and often have a lower turnaround time[10].

Scheduling Multiple-Level Queues

This method divides the ready queue into many queues. Processes are allocated to a queue in this technique depending on a certain attribute of the process, such as process priority, memory size, and so on.

However, this is not a stand-alone scheduling OS technique since it must employ other kinds of algorithms to schedule the processes.

Scheduling Multiple-Level Queues has the following characteristics:

• For operations with certain features, several queues should be maintained.

• Each queue may have its own scheduling method.

• Each queue is assigned a priority.

A Scheduling Algorithm's Function

Here are some of the benefits of employing a scheduling algorithm:

• Scheduling is used by the CPU to boost efficiency and to assist you distribute resources among competing activities.

• Multi-programming allows for maximum CPU usage; the tasks to be run are in a ready queue.

Scheduling Multiple Processors in the Operating System

A multi-processor system is one in which many processors share the same memory, bus, and input/output devices. The bus links the CPU to the RAM, I/O devices, and all other computer components.

The system has a high degree of coupling. This sort of system continues to function even if a processor fails. The remainder of the system continues to function normally. In multi-processor scheduling, many processors (CPUs) share the load to ensure that operations run smoothly.

Because of the following factors, the scheduling process of a multi-processor system is more difficult than that of a single processor system.

• Load balancing is a challenge since there are several processors present.

• Processes running concurrently may need access to shared data; scheduling should take cache affinity into account.

Methods for Scheduling Multiple Processors

Symmetric Multiprocessing: The processors self-schedule in symmetric multiprocessor scheduling. Each processor's scheduler scans the ready queue and chooses a process to run. Each processor runs the same copy of the operating system and interacts with one another. If one of the processors fails, the remainder of the system continues to function.

Global Queues for Symmetrical Scheduling: If the processes to be run are in a common or global queue, the scheduler for each processor examines the global-ready queue and chooses one to execute.

Symmetrical Scheduling with Individual Queues: If each processor in the system has its own private ready queue, the scheduler for that processor uses that private queue to choose a task.

Asymmetric Multiprocessing: A master server and slave servers are used in asymmetric multiprocessor scheduling. The master server controls all scheduling and I/O tasks, while the slave servers handle user processes. If the master server fails, the whole system grinds to a standstill. However, if one of the slave servers fails, the remainder of the system continues to function.

Processor Preference

A process has a preference for the processor on which it operates. This is known as processor affinity.

When a process runs on a processor, the most recent data accessed by the process is stored in the CPU's cache memory. The following data access calls made by the process are often met by cache memory; nevertheless, if this process is transferred to another processor for whatever reason, the content of the original processor's cache memory is invalidated, and the second processor's cache memory must be repopulated.

• Process migration from one processor to another is prevented to reduce the expense of invalidating and repopulating cache memory.

Processor affinity is classified into two sorts.

• Soft Affinity: The system contains a rule that attempts to keep a process running on the same processor but does not guarantee it. This is known as gentle affinity.

• Hard Affinity: The system enables the process to designate the subset of processors on which it may operate, i.e., each process can only run on some of the processors. Soft affinity is implemented in systems such as Linux, while hard affinity is supported through system methods such as sched setaffinity().

Load Distribution

All processors in a multi-processor system may not have the same workload. Some may have a lengthy ready line, but others may be inactive. Load balance comes into play to overcome this issue. Load balancing is the phenomena of spreading workload such that the processors in a symmetric multi-processor system have an equitable burden.

Load balancing is not necessary in symmetric multiprocessing systems with a global queue. A processor in such a system evaluates the global ready queue and picks a process as soon as it becomes optimal.

However, in an asymmetric multi-processor with private queues, some processors may be idle while others are busy. There are two options for dealing with this.

• Push Migration: During push migration, a job monitors the load on each processor on a regular basis. Some processors may have lengthy queues, while others may be idle. If the workload is unevenly divided, the burden will be extracted from the overloaded CPU and assigned to an idle or less busy processor.

• Pull Migration: A pull migration occurs when an idle CPU extracts the load from an overloaded processor.

Processors with many cores

A multi-core processor is a single computer component that is made up of two or more CPU cores. Each core has a register set to retain its architectural state and so appears as a distinct physical processor to the operating system. A CPU register may store an instruction, an address, or other data. Because each core contains a set of registers, the system operates as a multi-processor, with each core acting as a processor.

Symmetric multiprocessing systems with multi-core processors provide increased performance while using less energy.

Multiprocessor that is symmetric

Symmetric multi-processors have a single operating system that may be executed by any central processing unit. When a system call is made, the CPU that made the system call traps the kernel and processes it. The model dynamically balances processes and memory. It schedules processes using symmetric multiprocessing, as the name implies, and each processor is self-scheduling. Each processor searches the global or private ready queue for a process to run.

The kernel is the most important component of the operating system. It interfaces between the system hardware and the application software.

A symmetric multi-processor system may experience three types of conflict. These are listed below.

• Locking system: In a multi-processor, resources are divided across the processors. A locking mechanism is necessary to protect the processors' access to these resources. This is done to serialise the processors' access to resources.

• Shared data: Because numerous processors access the same data at the same time, the data may not be consistent across all of them. To prevent this, we must use some kind of mechanism or locking method.

• Cache Coherence: If resource data is stored in many local caches and shared by numerous clients, it may become invalid if one of the clients modifies the memory block. This can be fixed by keeping a consistent view of the data.

Multiprocessor with Master-Slave Mode

One CPU serves as the master in a master-slave multi-processor, while the others serve as slave processors. This implies that the master processor is in charge of all scheduling and I/O tasks, while the slave processors are in charge of user processes. Memory and I/O devices are shared by all CPUs, and all processors are linked to a single bus. It schedules processes via asymmetric multiprocessing.

Threading and virtualization

The technique of running several operating systems on a computer system is known as virtualization. As a result, a single CPU may also function as a multiprocessor. This may be accomplished by using a host operating system as well as various guest operating systems.

• Different apps operate without interfering with one another on different operating systems.

• A virtual machine is a virtual environment constructed on a physical hardware device that works as a virtual computer with its CPU, memory, network interface, and storage.

• A time-sharing operating system allocates 100ms (milliseconds) to each time slice to provide users with a fair response time. However, it takes more than 100ms, maybe a second or more. As a consequence, users registered onto the virtual machine experience slow response times. Because virtual operating systems only get a portion of the available CPU cycles, virtual machine clocks may be wrong. This is due to the fact that their timers do not take any longer to activate than those on dedicated CPUs.

Real-time systems are those that do tasks in real time. These duties must be completed as soon as possible and with a high level of urgency. These duties are specifically devoted to controlling (or responding to) certain occurrences. Real-time tasks are divided into two types: hard real-time tasks and soft real-time activities.

A difficult real-time work must be completed at a certain time, or else massive losses would occur. A deadline may be missed in soft real-time jobs. This is due to the fact that the work might be rescheduled (or) performed after the set time.

The scheduler, which is often a short-term job scheduler, is regarded as the most crucial component in real-time systems. Instead of dealing with the deadline, the major goal of this scheduler is to lower the response time connected with each of the linked activities.

If a preemptive scheduler is employed, the real-time task must wait until the time slice for its related job is completed. Even if the greatest priority is assigned to the job in the case of a non-preemptive scheduler, it must wait until the current task is completed. This job may be sluggish (or) of low priority, resulting in a lengthier delay.

Combining preemptive and non-preemptive scheduling creates a superior method. In priority-based systems, this may be accomplished by adding time-based interruptions, which implies that the presently running process is interrupted on a time-based interval, and if a higher priority process is available in a ready queue, it is performed by preempting the current process.

The scheduling algorithms are classed as follows based on schedulability, implementation (static or dynamic), and analysis outcome (self or dependent).

1. Static table-driven approaches: These algorithms typically do a static scheduling analysis and capture the most favourable schedules. This helps in creating a schedule that may identify a job whose execution must begin at run time.

2. Static priority-driven preemptive approaches: Like the previous technique, these algorithms employ static scheduling analysis. Instead than picking a specific timetable, it offers a handy method of allocating priority among numerous activities in preemptive scheduling.

3. Dynamic planning-based approaches: The workable schedules are identified dynamically in this technique (at run time). It has a predetermined time interval, and a procedure is conducted if and only if the time restriction is met.

4. Dynamic best effort techniques: These approaches prioritise deadlines above practical timelines. As a result, if the deadline is reached, the job is aborted. This strategy is commonly employed in most real-time systems.

REFERENCES

[1] S. Georgiou and D. Spinellis, "Energy-Delay investigation of Remote Inter-Process communication technologies," J. Syst. Softw., 2020, doi: 10.1016/j.jss.2019.110506.

[2] H. Dinari, "Inter-process communication (IPC) in distributed environments: An investigation and performance analysis of some middleware technologies," Int. J. Mod. Educ. Comput. Sci., 2020, doi: 10.5815/ijmecs.2020.02.05.

[3] L. Herrmann, M. Küttler, T. Stumpf, C. Baier, H. Härtig, and S. Klüppelholz, "Configuration of inter-process communication with probabilistic model checking," Int. J. Softw. Tools Technol. Transf., 2019, doi: 10.1007/s10009-019-00536-0.

[4] B. Shafabakhsh, R. Lagerström, and S. Hacks, "Evaluating the impact of inter process communication in microservice architectures," in CEUR Workshop Proceedings, 2020.

[5] A. Venkataraman and K. K. Jagadeesha, "Evaluation of inter-process communication mechanisms," Architecture, 2015.

[6] X. Geng, X. Zeng, L. Hu, and Z. Guo, "An Novel Architecture and Inter-process Communication Scheme to Adapt Chromium Based on Docker Container," in Procedia Computer Science, 2017. doi: 10.1016/j.procs.2017.03.149.

[7] A. Zafar and A. H. Akber, "Diagnostic Agent Based Inter-Process Communication Aware Monitoring System for Wireless Sensor Networks," Mehran Univ. Res. J. Eng. Technol., 2019, doi: 10.22581/muet1982.1902.07.

[8] L. Alawneh, A. Hamou-Lhadj, and J. Hassine, "Segmenting large traces of inter-process communication with a focus on high performance computing systems," J. Syst. Softw., 2016, doi: 10.1016/j.jss.2016.06.067.

[9] F. S. Gharehchopogh, E. Amini, and I. Maleki, "A new approach for inter process communication with hybrid of message passing mechanism and event based software architecture," Indian J. Sci. Technol., 2014, doi: 10.17485/ijst/2014/v7i6.6.

[10] M. S. Ahmed, J. Houser, M. A. Hoque, R. Raju, and P. Pfeiffer, "Reducing inter-process communication overhead in parallel sparse matrix-matrix multiplication," Int. J. Grid High Perform. Comput., 2017, doi: 10.4018/IJGHPC.2017070104.

CHAPTER 7

THREADS IN OPERATING SYSTEM (OS)

Dr C Kalaiarasan, Professor & Asso Dean
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- kalaiarasan@presidencyuniversity.in

A thread is a single combination of both iterative of tasks that are completed in a process; hence, it is also known as an execution thread or a control thread. Thread execution is possible inside the processes of any operating system. Aside from that, a process may have several threads. A distinct programme counter and a stack of activation records and control blocks are used by each thread of the same process. Thread is often described as a lightweight procedure[1].

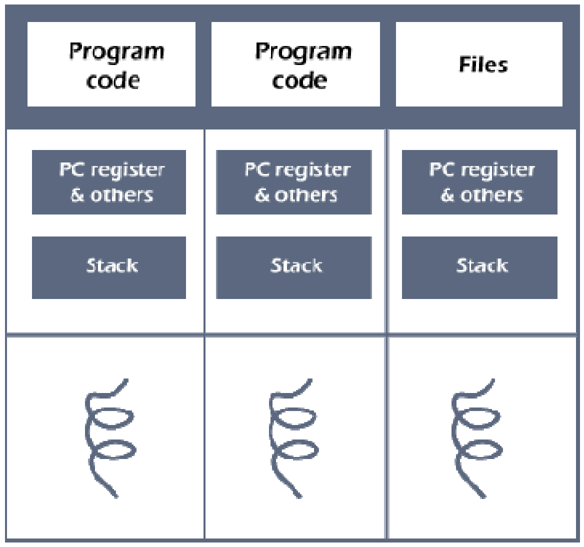


Figure: 7.1: Three threads of same process

The procedure may be divided into several threads. Many tabs in a browser, for example, may be considered as threads. MS Word employs several threads, with one thread formatting text and another thread processing input.

Need for Thread:

Creating a new thread in an existing process requires considerably less time than creating a new process.

Threads may share common data without requiring inter-process communication.

When dealing with threads, context switching is quicker.

Terminating a thread requires less time than terminating a process.

Thread Varieties

There are two kinds of threads in the operating system.

1. A thread at the kernel level.

2. A thread at the user level.

Thread at the user level

The user-level thread is not recognised by the operating system. User threads are simple to implement and are done by the user. The whole process is halted if a user executes a user-level thread blocking action. The kernel level thread has no knowledge of the user level thread. User-level threads are managed by the kernel-level thread as if they were single-threaded processes[2].

The Benefits of User-Level Threads

1. User threads are easier to construct than kernel threads.

2. User-level threads may be used with operating systems that do not support kernel-level threads.

3. It is more efficient and speedier.

4. Context switching takes less time than kernel-level threads.

5. It does not need any changes to the operating system.

6. The depiction of user-level threads is relatively basic. The address space of the user-level process contains the register, PC, stack, and micro thread control blocks.

7. It is straightforward to generate, switch, and synchronise threads without the process's interaction[3].

User-level Threads Have Drawbacks

1. There is no coordination between the thread and the kernel in user-level threads.

2. If a thread creates a page fault, the whole operation is halted.

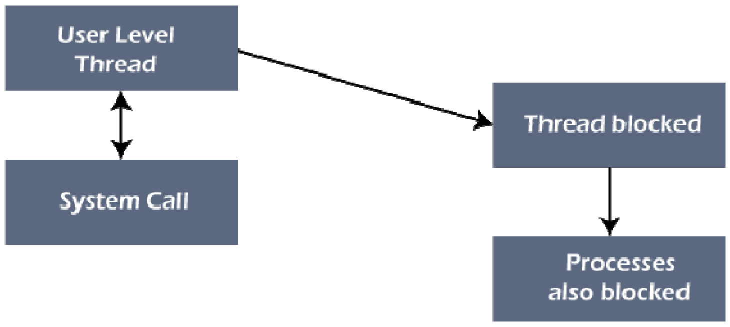


Figure: 7.2: Benefits and Drawbacks of User-Level Threads

Thread at the kernel level

The kernel thread is responsible for recognising the operating system. Each thread and process in the kernel-level thread has its own thread control block and process control block in the system. The operating system implements the kernel-level thread. The kernel is aware of all threads and handles them. To construct and manage threads from user-space, the kernel-level thread provides a system call. Kernel thread implementation is harder than user thread implementation. The kernel thread has a longer context switch time. If a kernel thread performs a blocking action, the execution of the Banky thread may continue. Solaris is an example of a window[4].

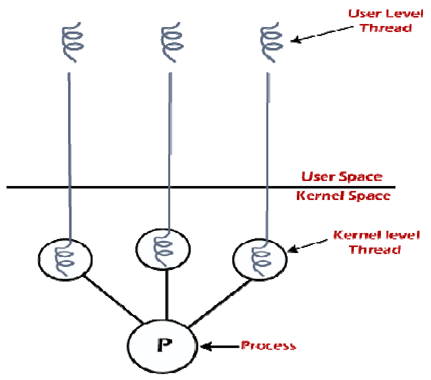


Figure: 7.3: Kernel-Level Threads

The Benefits of Kernel-Level Threads

1. All threads are completely aware of the kernel-level thread.

2. The scheduler may opt to spend more CPU time in the operation of big numerical threads.

3. The kernel-level thread is useful for apps that obstruct the frequency.

Kernel-level thread disadvantages

1. All threads are managed and scheduled by the kernel thread.

2. Kernel thread implementation is more complicated than user thread implementation.

3. User-level threads are slower than kernel-level threads.

Thread Components

Any thread contains the following elements.

1. The programme counter

2. Register configuration

3. Stacking space

Threads' Advantages

1. Increased system throughput: When the process is divided into several threads, each of which is handled as a job, the number of jobs completed per unit time rises. As a result, the system's throughput rises.
2. Effective Multiprocessor System Utilization: When you have several threads in one process, you may schedule multiple threads in multiple processors.

3. Faster context switching: The thread context switching duration is shorter than the process context switching period. The process context transition adds to the CPU's overhead.
4. Responsiveness: When the process is divided into threads, when one thread completes its execution, the other thread may be reacted to as quickly as feasible.
5. Communication: Multiple-thread communication is straightforward since the threads share the same address space, but in process communication, we utilise just a few exclusive communication mechanisms.
6. Resource sharing: Resources like as code, data, and files may be shared across all threads inside a process. It should be noted that the stack and register cannot be shared by several threads. Each thread has its own stack and register[5].

Multithreading

Multithreading is the execution of several threads at the same time. To grasp the notion of multithreading, you must first comprehend what a thread and a process are.

A process is a running programme. A process may be further subdivided into threads, which are sub-processes. For further information, please see the prior blog. Multithreading is shown by simultaneously playing a movie and downloading it.

There are two kinds of threads: user-level threads and kernel-level threads. So, in order for these threads to work together, they must have a connection. Multithreading Models are used to build this relationship. This link may be established in three ways.

Model of Many-to-One

Model of One-to-One

Model of Many-to-Many

Model of Many-to-One

Threads have a many to one connection, as the name implies. Multiple user threads are connected or mapped with a single kernel thread in this case. Thread management is done at the user level, which makes it more efficient.

Drawbacks

Because many user threads are assigned to a single kernel thread. So, if one user thread makes a blocking system call (such as function read(), the thread or process must wait until the read event is finished), it will block the kernel thread, which will then block all other threads.

Multiple threads cannot run in parallel on a multiprocessor system because only one thread may reach the kernel thread at a time. Even if we have several processors, one kernel thread will only execute on one of them. As a result, the user thread will only execute in the processor where the mapped kernel thread is operating.

Model of One-to-One

We may deduce from the name that one user thread is linked to one kernel thread.

Benefits of the Many-to-One Model

The first disadvantage of the Many-to-One approach is addressed in this model. Because each user thread is mapped to a distinct kernel thread, any user thread that performs a blocking system call will not block the other user threads.

The second disadvantage has also been overcome. It enables threads to execute in parallel on a multiprocessor. Because each kernel thread is assigned to a single user thread. As a result, each kernel thread may execute on a distinct CPU. As a result, each user thread may execute on one of the CPUs.

Disadvantages

Every time a user thread is created, a kernel thread must be created as well. As a result, the overhead of starting a kernel thread might have an impact on application performance.

In addition, the number of threads that may execute concurrently in a multiprocessor system is limited. If the system has four processors, then only four threads can execute at the same time. So, if you have 5 threads and attempt to run them all at once, it may not work. As a result, the programme should limit the number of kernel threads it supports[6].

Model of Many-to-Many

We may deduce from the name that the numerous user threads are mapped to a lower or equal number of kernel threads. The number of kernel threads is unique to each programme or computer.

Benefits over the other two models

Other threads are not stopped when a user thread performs a blocking system call.

There may be an unlimited number of user threads. Threads may also operate in parallel on many CPUs.

We don't need to construct as many kernel threads as we did for the user thread. As a result, there is no issue with any additional overhead imposed by the creation of a kernel thread.

The number of kernel threads supported in this case is determined by the programme or machine.

So, in a multithreading system, this is the optimal paradigm for establishing the link between user-thread and kernel thread.

Benefits of Multi-Threading Resource Sharing: Because threads may share any process's memory and resources, any programme can execute numerous tasks inside the same address space.

Utilization of Multiple Processor Architecture: Because distinct threads may run simultaneously on many processors, the processor can be used to its full potential and efficiency.

Reduced Context Switching Time: As with Thread Context Switching, the threads decrease the context switching time while maintaining the same virtual memory area.

Economic: Allocating memory and resources during process formation has a cost. Context-switch threads are more cost-effective because they can spread process resources.

OS Threading Issues

When we are in a multithreading environment, we have many threaded concerns. This section will go through threading difficulties such as system calls, thread cancellation, signal handling, thread pools, and thread-specific data.

Along with threading concerns, we will explore how these issues may be dealt with or resolved in order to continue to benefit from the multithreaded programming environment.

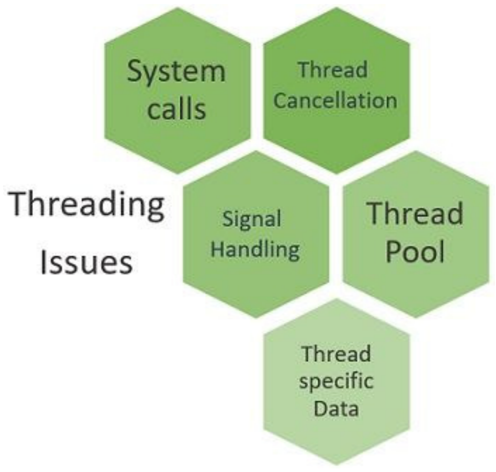


Figure: 7.3: OS Threading Issues

OS Threading Issues

1. System Inquiries
2. Cancellation of a Thread
3. Signal Processing
4. Pool of Threads
5. Thread-Specific Information

1. System calls fork() and exec()

The system calls are fork() and exec(). The fork() function makes a copy of the process that called fork (). The new duplicate process is known as the child process, while the process that used fork() is known as the parent process. Both the parent and child processes resume execution from the instruction immediately after the fork ().

Let us now look at the problem with the fork() system function. Consider the case when a thread of the multithreaded application has called the fork (). As a result, fork() would generate a new duplicate process. The question here is whether the new duplicate process produced by fork() would replicate all of the threads of the parent process or will be single-threaded.

In certain UNIX systems, there are two variants of fork(). Either fork() may duplicate all of the parent process's threads in the child process, or fork() will only duplicate the thread from the parent process that has called it[7].

The programme determines which version of fork() must be used.

When the exec() system call is executed, it replaces the programme and all its threads with the programme supplied in the exec() argument (). Normally, the exec() system call comes after the fork() system call.

The problem is that if the exec() system call comes just after the fork() system call, then forking all the threads of the parent process into the child process is pointless. Because the exec() system function replaces the whole process with the process specified in the argument.

In this scenario, the fork() variant that replicates just the thread that called fork() would be suitable.

2. Cancellation of a thread

Thread cancellation occurs when a thread is terminated in the midst of its execution. Let us look at an example to better grasp this. Consider a multithreaded software that has allowed its numerous threads to search a database for information. If one of the threads provides the required result, the other threads are cancelled.A thread that we intend to cancel is now referred to as the target thread. There are two methods for cancelling a thread:

Asynchronous Cancellation: In asynchronous cancellation, a thread is used to immediately kill the target thread.

Postponed Cancellation: In deferred cancellation, the target thread is scheduled to check whether it can terminate itself at regular intervals. The following issues are connected to the target threads:

• What if the resources were sent to the cancel target thread?

• What if the target thread is killed while changing data that it was sharing with another thread?

The asynchronous cancellation of the thread, in which a thread cancels the target thread without first verifying whether it is retaining any resources or not, causes problems.

However, with postponed cancellation, the thread that communicates the cancellation to the target thread, the target thread crosschecks its flag to determine whether it should be cancelled immediately or not. Thread cancellation occurs when they may be safely cancelled; such locations are referred to as cancellation points by Pthreads[8].

3. Signal Processing

Signal handling is more straightforward in a single-threaded application since the signal is passed directly to the process. However, when it comes to multithreaded programmes, the question of which thread of the programme the signal should be given to arises.

Assume the signal is sent to: all threads in the process.

Certain particular threads in the process. The thread to whom it relates.

Alternatively, you may allocate a thread to receive all of the signals.

Depending on the sort of produced signal, how the signal is transmitted to the thread is determined. The produced signal is of two types: synchronous signal and asynchronous signal.

Synchronous signals are routed via the same mechanism that generates the signal. Asynchronous signals are created by an event that occurs outside of the operating process, and the signals are received asynchronously by the running process.

So, if the signal is synchronous, it will be transmitted to the thread that generated the signal. If the signal is asynchronous, it is not possible to specify which thread of the multithreaded application it will be sent to. If the asynchronous signal indicates that the process should be terminated, the signal is sent to all threads in the process.

Most multithreaded UNIX systems address the problem of an asynchronous signal to some degree. The thread is given the option of specifying which signals it may and cannot take. However, the Windows operating system does not support the signal notion; instead, it employs asynchronous procedure call (ACP), which is analogous to the UNIX system's asynchronous signal.

The thread may select which signals it will take and which it will reject under UNIX, while the ACP is transmitted to the specified thread.

4. Pool of Threads

When a user requests a website from the server, the server starts a new thread to handle the request. However, the server may have some difficulties. Consider this: if there is no limit on the number of active threads in a system and we generate a new thread for each new request, we will eventually exhaust system resources.

We're also worried about the amount of time it will take to start a new thread. It must not be the case that the time necessary to establish a new thread exceeds the time required for the thread to fulfil the request before being discarded, since this would result in CPU time being wasted.

The thread pool is the answer to this problem. When the procedure begins, a limited number of threads are created. The thread pool is the name given to this group of threads. The threads remain in the thread pool until they are allocated a request to serve.

When a request arrives at the server, it starts a thread from the pool and allocates the request to it. The thread finishes its service and returns to the pool to await the next request.

If the server gets a request and there are no threads available in the thread pool, it waits for one of the threads to become free and return to the pool. This is preferable to starting a new thread for each request and is more comfortable for the system, which cannot manage a huge number of concurrent threads [9].

5. Thread-specific information

We are all aware that threads belonging to the same process exchange data from that process. The problem here is that each thread in the process need its own copy of the data. Thread-specific data refers to the particular data linked with a given thread.Consider a transaction processing system, where each transaction may be processed in a separate thread. To distinguish each transaction, we will assign it a unique identity. This will assist the system in uniquely identifying each transaction.

OS Process Synchronization

An operating system is a piece of software that manages all of the programmes loaded on a computer or other device, allowing it to work more effectively. As a result, the operating system usually needs to perform many tasks at the same time. Unless these processes are sharing a resource, this normally does not constitute a synchronisation issue in the operating system.

Consider a bank, which keeps a single database for each customer's account balance. Assume you have x rupees in your account at the start. Now you remove money from your bank account as someone else attempts to check your account balance[10].

REFERENCES

[1] K. Sharma, H. Saini, and G. Mehta, "Threads in Operating System," Int. J. Res. Appl. Sci. Eng. Technol., 2014.

[2] S. S. Bamber, "Implementation of improved synchronization in inter process communication using threads for microkernel and distributed operating systems," Int. J. Recent Technol. Eng., 2019, doi: 10.35940/ijrte.B1137.0982S1019.

[3] I. Akturk and O. Ozturk, "Adaptive Thread Scheduling in Chip Multiprocessors," Int. J. Parallel Program., 2019, doi: 10.1007/s10766-019-00637-y.

[4] O. N. Karasik and A. A. Prihozhy, "ADVANCED SCHEDULER FOR COOPERATIVE EXECUTION OF THREADS ON MULTI-CORE SYSTEM," «System Anal. Appl. Inf. Sci., 2017, doi: 10.21122/2309-4923-2017-1-4-11.

[5] Y. Qiu, P. Xiong, and T. Zhu, The Design and Implementation of the RT-Thread Operating System. 2020. doi: 10.1201/9781003099000.

[6] R. Grimm, "Multithreading," in Modernes C++: Concurrency meistern, 2018. doi: 10.3139/9783446456655.003.

[7] S. K. Jawad, R. P. Uhlig, B. Sinha, M. N. Amin, and P. P. Dey, "Multithread Affinity Scheduling Using a Decision Maker," Asian J. Comput. Inf. Syst., 2018, doi: 10.24203/ajcis.v6i4.5430.

[8] A. Castello, R. M. Gual, S. Seo, P. Balaji, E. S. Quintana-Orti, and A. J. Pena, "Analysis of Threading Libraries for High Performance Computing," IEEE Trans. Comput., 2020, doi: 10.1109/TC.2020.2970706.

[9] E. Asyabi, E. Sharafzadeh, S. A. SanaeeKohroudi, and M. Sharifi, "CTS: An operating system CPU scheduler to mitigate tail latency for latency-sensitive multi-threaded applications," J. Parallel Distrib. Comput., 2019, doi: 10.1016/j.jpdc.2018.04.003.

[10] D. Muelas, J. Ramos, and J. E. L. De Vergara, "Assessing the Limits of Mininet-Based Environments for Network Experimentation," IEEE Netw., 2018, doi: 10.1109/MNET.2018.1700277.

CHAPTER 8

SYNCHRONIZATION HARDWARE

Dr. Jayanthi Kamalsekaran, Assistant Professor
Department of Computer Science and Engineering, Presidency University, Bangalore, India
Email Id- jayanthi.k@presidencyuniversity.in

OS Process Synchronization Work

Let us examine why we need Process Synchronization. For example, if a process1 attempts to read data from a memory location while another process2 attempts to alter data from the same place, the data received by process1 is likely to be inaccurate.

Several programme elements/sections:

- Entry Section: The entry Section determines when a procedure begins.
- Critical Section: The critical section permits and ensures that only one procedure modifies the shared data.
- Exit Section: The Exit section handles the input of other processes into shared data following the execution of one process.
- Remainder Section: The Remainder section contains the remaining portion of the code that was not previously classified.

Synchronization Requirements

A solution to the critical section issue must meet the following three requirements:

Mutual exclusion: If one process is operating in the crucial part, no other process should be running in that area at the same time.

Progress: If no processes are still in the critical part and additional processes are waiting to execute outside the critical area, any of the threads must be allowed to enter the critical section. Only those processes that are not running in the remaining portion will decide which process will enter the crucial section.

There will be no starvation: Starvation occurs when a process waits indefinitely for access to a vital area but never receives it. Bounded Waiting is another name for no starving.

A procedure should not have to wait indefinitely to reach the crucial portion.

When a process requests access to its critical section, there should be a restriction or limitation that specifies the number of other processes that may access the critical portion before it[1].

This process should be permitted to access the vital region once this bound is reached.

Process Synchronization Types in an Operating System

The kinds of synchronisation are as follows:

1. Separate Processes

An independent process is one that is carried out without interfering with or having an effect on another process. For instance, the process without any common databases, files, variables, and so on.

2. Collaborative Procedures

Different processes in a computer system may function as independent or collaborative processes inside the operating system. When a process is claimed to be independent, it means that it will not be affected by other processes operating on the system. Independent processes do not share data between processes. Any other process operating on the system, on the other hand, may have an influence on a collaborating process. Data is exchanged between collaborating processes[2].

Operating System Synchronization Important

1. Mutual exclusion: No other process should be allowed to operate in the crucial area while the present one is running.

2. Progress: Any thread may reach the crucial area if no processes are still running within it and additional processes are waiting to execute outside of it. Only processes that are not executing in the other part will determine which process enters the crucial area.

3. No starvation: A starved process waits interminably for access to the vital part but is never given the chance. No hunger is another term for bounded waiting.

It should not take an eternity for a procedure to reach the critical region.

There should be a limit or constraint on how many other processes may access the crucial portion before a process can seek access to its critical section.

Once this bound is achieved, this process should be allowed access to the vital region.

Critical Section Issue

A crucial section is a portion of code that a signal process can access at a specified moment in time. The part is made up of shared data resources that must be accessible by other programmes.

• The wait() method handles the crucial section entry, which is represented as P. ().

• The signal() function, shown as V, controls the escape from a crucial region ().

Only one process may be run in the crucial region. Other processes that are waiting to execute their important portion must wait until the present process finishes[3].

Critical Section Guidelines

All three rules must be enforced in the vital section:

• Mutual Isolation: Mutual exclusion is a form of binary semaphore that is used to regulate access to a shared resource. It features a priority inheritance method to prevent difficulties with extended priority inversion. At any one moment, only one process may run in its crucial part.

• Progress: This solution is utilised when there is no one in the crucial area and someone wishes to enter. Then, in a limited period, those processes that are not in their reminder section should select who should go in.

Bound Waiting: When a process requests to enter the critical section, the number of processes that may enter the crucial section is limited. As a result, when the limit is reached, the system must enable the process to enter its critical portion.

Algorithms for Hardware Synchronization

The issue of process synchronisation may be tackled using both software and hardware. One of the software solutions to the process synchronisation issue is the Peterson solution. The Peterson algorithm enables two or more processes to share a single-use resource without interfering with one other. The Hardware solution to the issue will be discussed in this post. The following is the hardware solution:

1. Check and Adjust
2. Swap
3. Lock and Unlock

Set and Test

The test and set technique makes use of a boolean variable called 'lock,' which is initially set to false. This lock variable controls the process's entrance into the vital part of the code. Let's look at the algorithm first and then attempt to figure out what it's doing.

Swap

The swap function makes use of two boolean variables, lock and key. Both the lock and key variables are originally set to false. The lock and set algorithms are the same as the swap algorithm. When a process reaches the crucial area of the programme, the Swap algorithm utilises a temporary variable to set the lock to true[4].

Lock and unlock

To regulate the value of the lock, the unlock and lock algorithm use the TestAndSet function. For each process I the unlock and lock method employs a variable waiting[i]. Here, I is a positive number, such as 1,2,3,..., and corresponds to processes P1, P2, P3,..., and so on. waiting[i] determines whether or not the process I is waiting to reach the crucial section.

Before approaching the crucial part, all processes are kept in a ready queue. The processes are added to the queue in the order in which they were created. The circular queue is the queue.

Operating System Mutex

Mutex lock in an operating system is simply a binary variable that offers code-wise capabilities for mutual exclusion. At times, numerous threads may be attempting to access the same resource, such as memory or I/O. To ensure that no overriding occurs. Mutex functions as a locking mechanism.

Only one thread may acquire ownership of a mutex and apply the lock at the same time. Once the resource has been used, the mutex lock may be released.

Mutex use

A mutex offers mutual exclusion, allowing either the producer or the consumer to hold the key (mutex) and continue with their task. The user must wait as long as the producer fills the buffer, and vice versa. In a Mutex lock, only one thread may operate with the full buffer at any one moment.

When a programme begins, it asks that the system construct a mutex object for a certain resource. The system assigns a unique name or ID to the mutex object. When a programme thread wishes to access a resource, it takes the lock on the mutex object, uses the resource, and then releases the lock on the mutex object. The following process is then permitted to obtain the lock on the mutex object.

Meanwhile, a process has taken control of the mutex object, and no other thread or process may access it. If the mutex object is already locked, the process that wants to gain the lock on it must wait and is queued up by the system until the mutex object is unlocked[5].

Mutex's Benefits

- The mutex has the following advantages: o Mutex is merely a basic lock gained before entering its vital part and subsequently released.
- There are no race situations since only one thread is in its crucial region at any one moment, and data is always consistent.

Mutex's disadvantages

Mutex has various drawbacks, such as: o If a thread gains a lock and goes to sleep or is preempted, the other thread may not be able to proceed. This might result in famine.

- It cannot be locked or unlocked from a context other than the one in which it was obtained.
- In the important area, only one thread should be authorised at a time.
- The standard implementation may result in a busy waiting state, wasting CPU time.

OS semaphores

Mutex is a system for locking (waiting) and unlocking. Semaphores are signalling systems that inform processes about the condition of the Critical section in OS and provide access to it appropriately.

To control access to vital section code, semaphores use the following methods:

1. Wait

2. Signal

The Evolution of Semaphores in Operating Systems

The semaphore idea was developed by a Dutch computer scientist called Edsger Dijkstra and his colleagues while working on an operating system for the Electrologica X8. Over time, this technique became known as THE multiprogramming system. Dijkstra proposed a solution to the issue of wasting wake-up signals, which includes storing all wake-up calls. Instead of immediately giving the wake-up calls to the client, the producer, according to Dijkstra, may keep them in a variable. Any of the consumers may read it whenever they want.

The variable semaphore stores all of the wake-up calls issued from the producer to the consumer. It is a variable in kernel mode that is automatically modified, read, and updated.

A semaphore cannot be implemented in user mode because race situations are always possible when several processes attempt to access the variable at the same time. For implementation, it is always reliant on the operating system[6].

Semaphores of many types

Semaphores are classified as follows:

Semaphore Binary

Binary Semaphore is another term for Mutex lock. It has just two potential values: 0 and 1, and its default value is 1. It is used to implement a variety of techniques in order to tackle critical section issues. More information about Binary Semaphore may be found here.

Semaphore Count

Semaphore's value may be found everywhere in the globe. It is used to limit access to a resource that has many copies. More about Counting Semaphore may be found here.

Advantages of Semaphores

Some of the benefits of semaphores are as follows:

Because of semaphores, only one process is permitted to access the vital portion. They strictly adhere to the idea of mutual exclusion. They are also significantly more efficient than other synchronisation techniques.

Because the processor time is not spent monitoring whether a condition is satisfied to enable a process to access its crucial area, there is ultimately no resource waste due to busy waiting in the semaphores. As a result, they are machine-independent.

The Drawbacks of Semaphores

Some of the disadvantages of semaphores are as follows:

Because semaphores are complicated, the wait and signal operations must be implemented in the correct sequence to prevent deadlocks.

Semaphores are unsuitable for use at the ultimate size because they limit modularity. This is mostly due to the delay, as well as the signal protocols, preventing the system from developing an organised layout.

Semaphores may induce a priority inversion, with low-priority processes accessing the critical piece first and high-priority processes accessing it later[7].

The typical synchronisation issue

1) The bound buffer issue

It's also known as the producer-consumer issue. A bounded buffer enables numerous producers and consumers to share a single buffer. Producers write data to the buffer, while consumers read data from it. When the buffer is filled, producers must stop. When the buffer is empty, consumers must block.

2) The issue of the sleeping barber

It is a difficulty with synchronisation and inter-process communication. This issue is centred on barbershops. A barbershop has a single barber, a single barber chair, and n customer chairs. When there are no customers at the barbershop, the barber goes to sleep. When a client arrives, he must raise the barber. When there are numerous clients and the barber is cutting one of them's hair, the rest of the customers either wait for their number or leave when all the chairs are occupied.

3) The Dining Philosopher issue

In this classic synchronisation dilemma, five philosophers sit around a dinner table, doing nothing but eating and thinking. A bowl of noodles and five chopsticks are placed in the middle of the table. Only with both left and right chopsticks can a philosopher eat. When both left and right chopsticks are absent, the philosopher removes their chopsticks and resumes their reasoning.

4th. Readers and Writers Issue

The readers and writers dilemma is predicated on an object, like as a file, that is shared by several programmes. Some processes are readers, which means they can only read data, while others are writers, which means they can only write data. In most cases, this issue is employed to manage synchronisation. Semaphores may be used to solve this issue[8].

OS Critical Section Issue (Operating System)

A critical section is a section of a software that attempts to access shared resources. That resource might be any computer resource, such as a memory region, data structure, CPU, or any IO device.

The critical section cannot be performed by more than one process at the same time; the operating system has challenges in permitting and preventing processes from accessing the critical part.

The critical section issue is used to create a set of procedures that guarantee that the Race condition between processes never occurs.

Our key aim is to address the critical section issue in order to synchronise the cooperative processes. We must devise a solution that will allow the following requirements to be met.

Critical Section Problem Solutions

A crucial section solution should contain the following characteristics:

Once a process has entered the crucial area, no other process should be permitted to enter. This is known as mutual exclusion.

If a process is in the crucial section and another process arrives, the new process must wait until the previous process departs the critical area before proceeding. In such instances, the process that is waiting to access the vital part should not be kept waiting indefinitely. This is referred to as progress.

If a process wishes to enter the crucial part, there should be a time limit for the process to wait. This is known as bounded waiting.

The solution should be independent of the architecture of the system. This is referred to as neutrality.

Peterson's solution, semaphores, and monitors are examples of software-based solutions for critical section difficulties. Some hardware-based solutions to the critical section issue include atomic instructions like TestAndSet, compare and swap, Unlock and Lock.

Monitor

It is a synchronisation mechanism that allows threads to mutually exclude each other and wait() for a specific condition to become true. It is a kind of abstract data. It has a common variable and a set of operations that run on that variable. A process may not directly access the shared data variables, and protocols must be in place to enable several processes to access the shared data variables at the same time.

In a monitor, only one process may be active at any given moment. Other processes requiring access to the shared variables must wait and are allowed access only when the preceding process releases the shared variables[9].

Monitor Characteristics in OS

A monitor in os has the following features:

Within the monitor, we can only execute one application at a time. • Monitors in an operating system are defined as a combination of methods and fields that are coupled with a particular kind of package in the os.

If a programme is executing outside of the monitor, it cannot access the monitor's internal variable. However, a software may access the monitor's functionalities.

Monitors were established to simplify synchronisation concerns and to give a high degree of synchronisation between activities.

Monitor components in an operating system

The display is made up of four major components:

1. Initialization: The initialization code is supplied in the package, and it is only required once when generating the monitors.

2. Private Data: A feature of an operating system's monitor that makes data private. It contains all of the monitor's secret data, including hidden features that may be used only inside the monitor. As a consequence, confidential fields and functionalities are hidden behind the monitor.

3. Monitor Procedure: Monitor procedures are processes or services that may be called from outside the monitor.

4. Keep an eye on the entry queue: The Monitor Entry Queue is another critical component of the monitor. It only includes the threads that are typically referred to be procedures.

Variables of Conditions

We may conduct two types of operations on the monitor's condition variables:

1. Wait
2. Signal

Consider the following condition variable (y) that is defined in the monitor:

y.wait(): The activity/process that performs the wait operation on a condition variable is suspended, and the suspended process is placed in the block queue of the condition variable.

y.signal(): If an activity/process uses the signal action on the condition variable, one of the monitor's stalled activities/processes is given a chance to run.

Benefits of an OS Monitor

Monitors make concurrent or parallel programming simpler and less error-prone than semaphore-based solutions.

Why It aids in operating system process synchronisation.

Monitors have built-in mutual exclusion.

Monitors are simpler to set up than semaphores.

Monitors may be able to remedy for timing problems caused by semaphores.

Disadvantages of OS Monitor

Monitors must be written in a programming language. Monitors increase the strain on the compiler. The monitor must understand what operating system capabilities are available for regulating critical parts of parallel operations[10].

REFERENCES

[1] E. Guillo-Sansano, M. H. Syed, A. J. Roscoe, and G. M. Burt, "Initialization and synchronization of power hardware-in-the-loop simulations: A Great Britain network case study," Energies, 2018, doi: 10.3390/en11051087.

[2] A. D. Edelstein, M. A. Tsuchida, N. Amodaj, H. Pinkard, R. D. Vale, and N. Stuurman, "Advanced methods of microscope control using μManager software," J. Biol. Methods, 2014, doi: 10.14440/jbm.2014.36.

[3] L. Shen, X. Feng, Y. Zhang, M. Shi, D. Zhu, and Z. Wang, "Stroboscope based synchronization of full frame CCD sensors," Sensors (Switzerland), 2017, doi: 10.3390/s17040799.

[4] M. J. Colville, S. Park, W. R. Zipfel, and M. J. Paszek, "High-speed device synchronization in optical microscopy with an open-source hardware control platform," Sci. Rep., 2019, doi: 10.1038/s41598-019-48455-z.

[5] M. Bilucaglia et al., “ESB: A low-cost EEG Synchronization Box,” HardwareX, 2020, doi: 10.1016/j.ohx.2020.e00125.

[6] F. Li, L. Li, and G. Shi, “Multi-service oriented stream data synchronization scheme for multicore cipher chips ∗,” IEICE Trans. Fundam. Electron. Commun. Comput. Sci., 2019, doi: 10.1587/transfun.E102.A.48.

[7] L. Wang, Y. Dong, D. Xie, and H. Zhang, “Synchronization control of stochastic delayed Lotka–Volterra systems with hardware simulation,” Adv. Differ. Equations, 2020, doi: 10.1186/s13662-019-2474-9.

[8] A. Zhai, J. G. Steffan, C. B. Colohan, and T. C. Mowry, “Compiler and hardware support for reducing the synchronization of speculative threads,” Trans. Archit. Code Optim., 2008, doi: 10.1145/1369396.1369399.

[9] N. Dahasert, I. Öztürk, and R. Kiliç, “Experimental realizations of the HR neuron model with programmable hardware and synchronization applications,” Nonlinear Dyn., 2012, doi: 10.1007/s11071-012-0618-5.

[10] K. Bezas, V. Komianos, G. Koufoudakis, G. Tsoumanis, K. Kabassi, and K. Oikonomou, “Structural health monitoring in historical buildings: A network approach†,” Heritage, 2020, doi: 10.3390/heritage3030044.

CHAPTER 9

DEADLOCK IN AN OPERATING SYSTEM (OS)

Prof Neeraj Sharma, Professor
Department of SOCSA, IIMT University, Meerut Uttar Pradesh, India
Email id- talk2neerajsharma@yahoo.co.in

A deadlock occurs when two computer programmes that share the same resource effectively prohibit each other from accessing the resource, causing both programmes to stop working.The first computer operating systems only allowed one application to run at a time. This single software had access to all of the system's resources. Later, operating systems interleaved numerous programmes at the same time. Programs were forced to identify what resources they need in advance in order to prevent conflicts with other programmes that were executing at the same time. Eventually, several operating systems included dynamic resource allocation. After they had started operating, programmes might seek further resource allocations. This resulted in the deadlock situation.

Every process requires the use of resources in order to be completed. The resource, however, is provided in a sequential sequence.

1. The procedure seeks a resource.
2. If the resource is available, the OS should provide it; otherwise, the process should wait.
3. The procedure employs it and releases it upon completion.

A deadlock occurs when each computer process waits for a resource that is being allocated to another process. In this case, none of the processes get performed since the resource they need is being held by another process that is simultaneously waiting for another resource to be released.

Assume there are three processes P1, P2, and P3. There are three distinct materials available. R1, R2, R3, and R4. R1 has been allocated to P1, R2 has been assigned to P2, and R3 has been assigned to P3.

After some time, P1 requests R1, which is now in use by P2. P1 terminates its execution since it cannot continue without R2. P2 also requests R3, which is utilised by P3. P2 likewise comes to a halt since it cannot continue without R3. P3 also requests R1, which is being utilised by P1, and hence P3 suspends its execution[1].

Conditions Required for OS Deadlock

It is not required for a stalemate to occur if more than one process and more than one resource are involved. This is because a stalemate is defined by four requirements that must be met regardless. These are some examples:

1. Mutual Isolation

Mutually exclusive indicates that there is a specific resource for each process that cannot be shared. No two processes may now request the same resource.

2. Hold your breath and wait

This stalemate scenario in OS, as the name implies, requires a process to wait for an occupied resource. This condition could not be more accurate. Example of a Deadlock

3. There is no preemption.

At any one moment, only one process may be scheduled. As a result, it is argued that no two processes may run concurrently. This means that only after a procedure is done will the assigned or occupied resource be freed.

4. Waiting in a circle

Each group of processes waits in a cyclic fashion. This keeps them waiting indefinitely, and they are never executed. As a consequence, deadlocks are referred to as "circular waits" because they cause a process to get stuck in a cyclical pattern. Each group of processes waits in a cyclic fashion. This keeps them waiting indefinitely, and they are never executed. As a consequence, deadlocks are sometimes known as "circular waits" because they cause a process to get stuck in a cyclical pattern[2].

So those were the four stalemate circumstances.

Methods for Dealing with Deadlock in OS

1. Detection and Recovery of Deadlocks

This approach entails diagnosing the deadlock scenario and implementing various methods to recover your system from a stalemate. A user may simply terminate any processes that might lead to subsequent deadlocks. It is preferable to terminate one process at a time rather than all processes at once. Continue to terminate processes until the system recovers from the stalemate state. Another option is to release the resources from allocation until the impasse is broken. This is known as "resource preemption".

2. Avoiding Deadlock

A deadlock will be averted if a user stops any of the four prerequisites for a deadlock from occurring.

3. Avoiding Deadlock

When a process requests that a resource be allocated, an algorithm is used to determine whether or not this resource allocation is safe for the system. If the request is not safe, it is rejected. This algorithm is referred to as the "deadlock avoidance" approach.

Ignorance of the Deadlock

UNIX, Windows, and certain other operating systems are often unaware of deadlocks and dismiss the problem whenever it happens. The "Ostrich algorithm" refers to this strategy. So, anytime a stalemate develops, just restart your computer, and the impasse will be resolved in no time[3].

Model of the System

A system is made up of a limited quantity of resources that must be divided among many competing processes. The resources are divided into numerous kinds, each with a certain number of identical instances. Resource types include memory space, CPU cycles, files, and I/O devices (such as printers and DVD drives). When there are two CPUs in a system, the resource type CPU has two instances. Similarly, there may be five instances of the resource type printer. If a process asks an instance of a resource type, any instance of the type will be allocated to meet the request. If it does not, the instances are not identical, and the resource type classes have not been correctly declared.

A system, for example, may contain two printers. If no one cares which printer outputs which output, these two printers might be specified to be in the same resource class. However, if one printer is on the ninth floor and the other is in the basement, individuals on the ninth level may not consider both printers to be comparable, and separate resource courses for each printer may be required. A process must request a resource before utilising it and must release it once it has been used.

A process may request as many resources as it needs to complete its assigned job. Obviously, the total amount of resources available in the system cannot exceed the number of resources requested. In other words, if the system only has two printers, a process cannot request three. In its typical mode of operation, a process may only use a resource in the following order:

1. Make a request. If the request cannot be granted immediately (for example, if the resource is under use by another process), the requesting process must wait until the resource is available.

2. Use, The resource may be used by the process (for example, if the resource is a printer, the process can print on the printer).

3. Let go. The resource is released as a result of the procedure. As stated in Chapter 2, system calls are used to request and release resources. Request () and release() device calls, open() and close () file calls, and allocat e () and free () memory system calls are a few examples. The waitO and signal () actions on semaphores, or the acquisition and release of a mutex lock, may be used to request and release resources that are not maintained by the operating system. The operating system examines each usage of a kernelmanaged resource by a process or thread to ensure that the process requested and was allocated the resource.

A system table keeps track of whether each resource is free or assigned; for each resource that is assigned, the table also keeps track of the process to which it is assigned. If a process requests a resource that is already being used by another process, it might be added to a queue of processes that are waiting for this resource. When every process in a set is waiting for an event that can only be triggered by another process in the set, the set is said to be in a deadlock condition[4].

The key events we're concerned with here are resource acquisition and release. Physical resources (such as printers, tape drives, memory space, and CPU cycles) and logical resources are both possible (for example, files, semaphores, and monitors). Other sorts of events, on the other hand, may result in deadlocks (for example, the 1PC facilities discussed in Chapter 3). Consider a system with three CD RVV drives to demonstrate a stalemate situation. Assume that each of the three processes has one of these CD RW drives. If each process now demands another disc, the three processes will get stuck in a loop. Each is anticipating the event "CD RVV is released," which can only be triggered by one of the other waiting processes. This is an example of a stalemate using the same resource type. Different resource types may also be involved in deadlocks. Consider a system with a single printer and a single DVD drive.

Assume that process P. is in charge of the DVD and process P; is in charge of the printer. A stalemate arises if P asks the printer and P. demands the DVD drive. A programmer working on multithreaded programmes must pay close attention to this issue. Because numerous threads might compete for shared resources, multithreaded applications are prone to deadlock[5].

Characterization of Deadlock

1. Deadlock Conditions

2. Graph of System Resource Allocation

Prerequisites for Deadlock

I Mutual Isolation

In a multiprogramming environment, many processes may be demanding the same resource at the same time. The mutual exclusion condition restricts access to the resource to a single process at a time. Other processes needing the same resource must wait and postpone their execution until it is released. The mutual exclusion condition stops two or more processes from simultaneously accessing the same resource.

ii) Be patient and wait

The hold and wait condition simply indicates that the process must be holding access to one resource while waiting for other processes to obtain additional resources.

iii) There is no preemption.

A process that is acquiring a resource cannot be interrupted in the middle to release the obtained resource. Instead, when the process's work is accomplished, it must willingly relinquish the resource it has gained.

Circular Delay

To get the resource, the processes must wait in a cyclic pattern. The only difference between this and hold and waits is that the processes wait in a cyclic fashion.

Let me illustrate this with an example. P0, P1, P2, P3, and P4 are the five processes. Now, process P0 is waiting to acquire the process owned by P1, and process P1 is waiting to acquire the resource held by P2, and so on until process P4 acquires the resource held by P0.

Preventing Deadlocks in the Operating System

A process is a series of steps. When a process operates, it requires resources like as CPU cycles, files, or access to peripheral devices[6].

Some resource requests may result in a stalemate.

Deadlock prevention is removing one of the required criteria for deadlock, allowing only safe requests to be made to the OS and excluding the potential of deadlock before initiating requests.

Because requests are now made with care, the operating system can securely approve all requests.

The operating system does not need to do any extra work in this case, as it does in deadlock avoidance by performing an algorithm on requests to check for the risk of deadlock.

OS Deadlock Prevention

To prevent deadlocks, dynamic information such as the present state of the system and resource allocation are used. To avoid a deadlock from happening, static information such as the maximum amount of resources a process may request is used. For deadlock avoidance, resource allocation techniques such as the Banker's Algorithm may be utilised.

Deadlock prevention is a method used in operating systems to avoid situations in which two or more processes are unable to advance because one of them is waiting for the other to relinquish a resource. This may be accomplished by placing a set of limits on how the processes request and release resources[7], [8].

Avoiding and Preventing Deadlock

Some typical deadlock avoidance tactics include:

1. Allocating resources in a method that assures a process never enters a stalemate state, such as utilising the Banker's Algorithm, which checks for safe states before allocating resources.

2. Resource limitation: A stalemate situation may be avoided by restricting the quantity of resources a process can request.

3. Timeout: By specifying a timeout for resource requests, a process will relinquish a resource if it is unable to get it within a specified time frame.

4. Priority-based resource allocation: A stalemate condition may be avoided by allocating resources based on the process's priority.

5. Preemptive resource allocation: A stalemate condition may be avoided by preempting resources from a process that has been holding them for a long period.

These are some of the most frequent strategies for preventing deadlocks in operating systems. The optimal strategy to utilise is determined by the system's unique needs and the resources available.

OS deadlock avoidance

Deadlock avoidance is a strategy used in operating systems to avoid situations in which two or more processes are unable to advance because one of the others is waiting for one of the others to release a resource. Deadlock avoidance, as opposed to deadlock prevention, leverages dynamic information, such as the current state of the system and resource allocation, to guarantee that a deadlock never arises. Some typical strategies of avoiding stalemate include:

1. Wait/Die: In this strategy, a process that seeks an unavailable resource will wait until the resource becomes available. If a process that is holding the resource also requests a resource that is not available, the process that has been waiting the longest will be permitted to continue.

2. Wound/Wait: In this strategy, a process that seeks an unavailable resource is destroyed (wounded) if a process that holds the resource simultaneously asks an unavailable resource. The terminated process will have to request the resource again later.

3. Resource allocation graph: The operating system may employ algorithms to verify for safe conditions before allocating resources by portraying the resources and processes as nodes in a graph.

4. Banker's Method: A variation of the Resource Allocation Graph algorithm, this algorithm utilises the available and maximum resource information for each process, and it verifies for a safe condition before allocating resources.

5. Time-stamp ordering: A process may only request a resource if its time stamp is greater than the time stamp of the process that has the resource.

These are some of the most frequent strategies for avoiding deadlocks in operating systems. The optimal strategy to utilise is determined by the system's unique needs and the resources available[9], [10].

REFERENCES

[1] S. Moharrami and A. Karimov, “Deadlock avoidance in the operating systems with using coloured Petri nets,” Int. J. Soft Comput., 2012, doi: 10.3923/ijscomp.2012.38.43.

[2] H. Karcanaj, E. Bumci, I. Tafa, and J. Fejzaj, “Deadlocks in different operating systems,” in Proceedings - 12th International Conference on Information Technology: New Generations, ITNG 2015, 2015. doi: 10.1109/ITNG.2015.120.

[3] P. Shruthi and N. G. Cholli, “An analysis of software aging in cloud environment,” Int. J. Electr. Comput. Eng., 2020, doi: 10.11591/ijece.v10i6.pp5985-5991.

[4] J. Krystek and M. Kozik, “Analysis of the job shop system with transport and setup times in deadlock-free operating conditions,” Arch. Control Sci., 2012, doi: 10.2478/v10170-011-0032-0.

[5] T. PatanasakPinyo, “Model checking approach for deadlock detection in an operating system process-resource graph using dynamic model generating and computation tree logic specification,” in Proceedings of 34th International Conference on Computers and Their Applications, CATA 2019, 2019. doi: 10.29007/fmrb.

[6] A. W. Salim et al., “Implementation Resource Request Alghoritm in Simulation of Deadlock Avoidance,” in Journal of Physics: Conference Series, 2019. doi: 10.1088/1742-6596/1230/1/012096.

[7] S. Pang, H. Chen, H. Liu, J. Yao, and M. Wang, “A deadlock resolution strategy based on spiking neural P systems,” J. Ambient Intell. Humaniz. Comput., 2019, doi: 10.1007/s12652-019-01223-3.

[8] M. Devi B., S. Agrawal, and R. R. Rao, “Deadlock free resource management technique for iot-based post disaster recovery systems,” Scalable Comput., 2020, doi: 10.12694:/scpe.v21i3.1734.

[9] E. Aldakheel, U. Buy, and S. Kaur, “DDS: Deadlock Detector and Solver,” in Proceedings - 29th IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2018, 2018. doi: 10.1109/ISSREW.2018.00009.

[10] V. A. T. Manurung, Y. Trijoko, and R. P. Afani, “MENURUNKAN KERUSAKAN YANG TIDAK TERJADWAL (UNSCHEDULE BREAKDOWN) SISTEM BAHAN BAKAR PADA UNIT KOMATSU HD 1500-7 DIPT UT SITE KALIMANTAN TIMUR,” J. Mech. Eng. Mechatronics, 2019, doi: 10.33021/jmem.v4i1.660.

CHAPTER 10

MEMORY MANAGEMENT

Prof Neeraj Sharma, Professor
Department of SOCSA, IIMT University, Meerut Uttar Pradesh, India
Email id- talk2neerajsharma@yahoo.co.in

An operating system's ability to handle or manage primary memory and switch among main memory and the disc during the functioning of the system is known as memory management. Whether a memory address is free or has been allocated to a process, memory management maintains a record of all of them. It determines how much memory should be set aside for processes. It chooses which processes will receive memory and when. It keeps track of when memory is released or unallocated and updates the status accordingly[1].

Reasons Memory Management is Necessary:

1. Prior to and after a process's execution, allocate and release memory.
2. To monitor how much RAM various processes are using.
3. To reduce fragmentation-related problems.
4. To make optimal use of the main memory
5. To keep data integrity while a process is being executed.

Techniques for Memory Management:

Here are some of the most important memory management strategies:

1. **Single Contiguous Allocation:**

It is the simplest method of memory management. With the exception of a limited amount that is set aside for the operating system, all kinds of computer memory are made accessible for one application in this manner. For instance, the MS-DOS operating system does this while allocating RAM. A single application also runs on an embedded system.

2. Partitioned Allocation:

It partitions primary memory, which consists primarily of contiguous memory sections, into several memory partitions. Each division houses all the data necessary to complete a particular task or job. This approach involves allocating a division to a work at startup and leaving it unallocated at shutdown.

3. Paged Memory Management:

By using this technique, page frames of fixed sizes are used to partition the computer's main memory. This hardware memory management device divides pages into frames, which ought to be allotted on a page-by-page basis.

4. Segmented Memory Management:

The user's software is not given a linear and continuous address space by any memory management technique other than segmented memory. A segment table is required as hardware support for segments. It includes the section's physical address in memory, size, as well as other information like access protection flags and status.

The primary memory in a computer system is the primary storage unit. It has a big and fast associative memory and stores programmes and information throughout computer operations. The primary memory's technology is based on semiconductor integrated circuits[2].

RAM stands for random access memory. Integrated circuit Random Access Memory (RAM) chips may operate in two modes, which are as follows:

Static It is made up of internal flip-flops that hold binary information. When electricity is applied to the device, the stored data stays stable. Static RAM is easy to utilise and has less read and write cycles.

Dynamic It stores binary data in the architecture of electric charges utilised in capacitors. Metal Oxide Semiconductor (MOS) transistors within the device make the capacitors accessible. The stored value on the capacitors adds to discharge over time, hence the capacitors should be refilled on a regular basis by activating the dynamic memory.

RAM stands for Random Access Memory

RAM, or Random Access Memory, is a kind of memory that can be quickly read and written to by a microprocessor. To be considered random access memory, it must be able to access any address at any time. This distinguishes RAM from sequentially accessed storage devices such as cassettes or hard discs.

RAM is a computer's primary memory. Its purpose is to store presently used data and programmes. This memory is managed by the operating system. It specifies when the things should be loaded into RAM, where they should be placed in RAM, and when they should be deleted from RAM.

Memory That Can Only Be Read

Every computer system should have a set chunk of memory that is unaffected by power outages. This form of memory is referred to as Read-Only Memory (ROM).

SRAM

Static Random Access Memories are RAMs that are made up of circuits and can store information as long as power is given (SRAM). The fundamental memory components of an SRAM device are flip-flops. An SRAM is made up of a series of flip-flops, one for each bit. SRAM is made up of an array of flip-flops; more flip-flops are required to give larger capacity memory. As a result, simpler flip-flop circuits, BJT transistors, and MOS transistors are employed for SRAM.

DRAM SRAMs are quicker, but they are more expensive since their cells need a large number of transistors. RAMs may be produced at a reduced cost by using simpler cells. To replace the SRAM cells, a MOS storage cell based on capacitors may be employed. A storage cell of this kind cannot retain the charge (that is, data) forever and must be recharged on a regular basis. As a result, these cells are known as dynamic storage cells. RAMs that use these cells are known as Dynamic RAMs, or simply DRAMs.

Obviously, memory accesses and memory management are critical components of contemporary computer functioning. Before an instruction can be performed, it must be obtained from memory, and most instructions require obtaining data from memory, storing data in memory, or both.

The introduction of multi-tasking operating systems has increased the complexity of memory management since, when programmes are switched in and out of the CPU, their code and data must be swapped in and out of memory, all at high rates and without interfering with other processes.

Shared memory, virtual memory, memory categorization as read-only vs read-write, and notions such as copy-on-write forking all complicate matters[3].

Swapping

Swapping is the process of shifting data from physical memory (RAM) to secondary memory. Virtual memory is a management strategy in computing that combines a computer's hard disc space with its random access memory (RAM) to generate a bigger virtual address space. This might be handy if your system has too many processes running and not enough physical memory to hold them.

The operating system must allocate a block of memory during swapping. It locates the first empty block of physical memory.

Each new memory block allocated replaces the oldest block in physical memory. When a programme tries to access a swapped-out page, the operating system transfers the page from disc into physical memory and changes the page table entry.

The goal of operating system shifting is to improve multiprogramming and enhance main memory use[3].

Changing the operating system consists of two steps:

Swap-in: The process of moving a process from secondary storage / hard drive to main memory (RAM). Swap out: Swap out transfers a process from main memory to secondary memory.

Operating system switching is required

The primary goal of memory management swapping is to provide more accessible memory than the computer hardware. Physical memory may be allocated, and the process may need the allocation of extra memory. Memory shifting, as opposed to forcing the system to utilise just memory based on physical RAM, enables the operating system and its users to extend memory to disc.

Benefits of Swapping

Swapping may help free up space and make your applications function more smoothly. Furthermore, swap files might be useful when you have numerous apps running at the same time.

A swap file allows you to guarantee that each application has its own dedicated block of memory, which may increase overall performance.

Increase the degree of multi-programming; improve RAM consumption.

The Drawbacks of Swapping

If the computer system is shut off during a period of excessive paging activity, the user may lose all program-related data.

As the number of page faults grows, overall processing speed suffers. As the number of transactions increases, users lose information and computers lose power[4].

The Swapping method in the Operating System

Swapping is the interchange of processes. Priority-based preemptive scheduling is also used in process interchange. When a higher priority process comes, the memory management temporarily switches out the lowest priority process to disc, and the highest priority process is swapped in for execution in main memory. When the highest priority process exits, the lower priority process returns to memory and continues to execute. This is referred to as switching.

Because of the address binding approach, processes that are paged out of main memory share the same address space when they are paged back in if the binding happens during load time. If it has run-time binding, the address is determined at run time, allowing the process to access any address space in main memory[5].

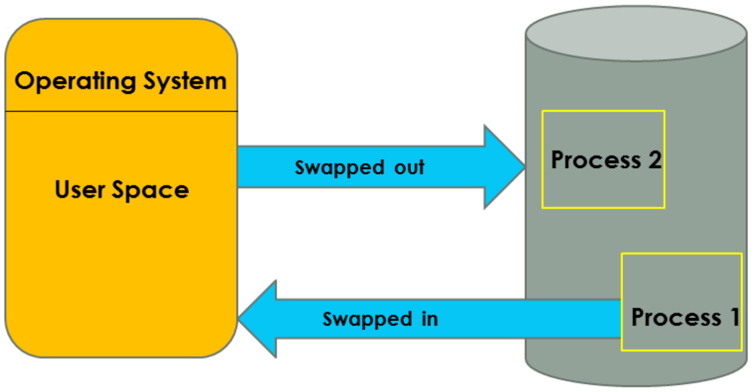


Figure 10.1: Swapping Method in Operating System

When the process is switched in and out, it leaves many memory holes. The holes are joined to form a large memory space. This is accomplished by shifting all processes as far down as feasible. This is referred to as compaction. However, this takes a significant amount of CPU time. As a result, this approach is seldom employed. The available free space is 40+20+20 K, as shown in the picture below. This open area is not connected in any manner. As a result, this open space is compressed to produce 80 K space.

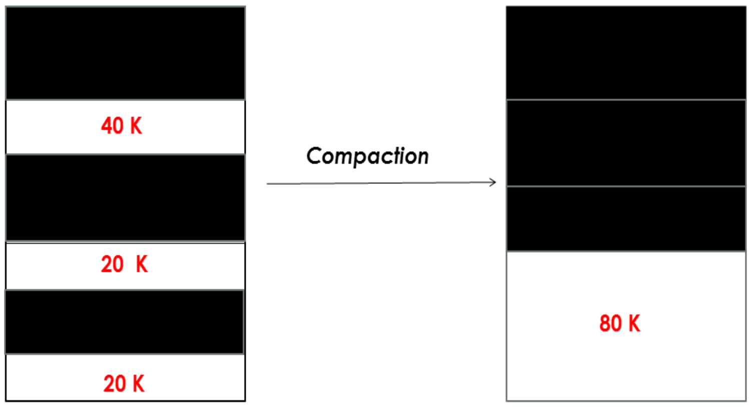


Figure 10.2: File and space swapping

File and space swapping

When physical space has been depleted, swap space is employed as a temporary storage capacity. A paging file is a physical disc storage file used by the operating system to extend available RAM. Because the operating system reuses physical memory, paging files and related memory pages may be regenerated in various locations of the system's virtual memory over time.

Purpose of swap space

Swap space assists in making your computer's operating system seem to have more RAM than it really has. A paging file is another name for a paging file. This data movement between virtual and physical memory is referred to as swap space, and disc space is referred to as "swap space."

Contiguous Memory Allocation

It is a way of allocating memory. When a process requests memory, a single contiguous piece of memory blocks is assigned based on its needs.

It is finished by dividing the memory into fixed-size divisions and allocating each partition to a particular process. It will, however, restrict the degree of multiprogramming to the number of fixed partitions performed in memory.

Internal fragmentation is also caused by this allocation. Assume a process is allotted a fixed-sized memory block that is somewhat larger than its requirement. Internal fragmentation refers to the leftover memory space in the block in this instance. When a process in a partition completes, the partition becomes available for use by another process.

In the variable partitioning scheme, the operating system keeps a table that shows which memory partitions are free and which are occupied by processes. By reducing address translation overheads, contiguous memory allocation speeds up process execution[6].

Process of Contiguous Memory Allocation

To begin, you must realise that a computer's memory must handle both the operating system and user processes. Contiguous memory allocation often splits existing memory into two parts: one for the operating system and one for all user processes or other applications. It does not split operating systems or user processes by software, but rather groups them into adjacent blocks given to each large process category.

Contiguous memory allocation is often implemented by dividing the memory into many fixed-sized parts. Each component may be assigned to a single process. The capacity of the computer to execute several applications is therefore determined by the number of partitions. Only when one part is available for execution may another process from the input queue be loaded. This approach was originally utilised by the IBM OS/360 operating system but is no longer used[7].

Advantages of Contiguous Memory Allocation

1. It allows a user to access files at random.

2. The user experiences exceptional read performance.

3. It is rather easy to apply.

Cons of Contiguous Memory Allocation 1. It may be difficult to have a file expand.

2. The disc might be fragmented.

Techniques for Allocating Contiguous Memory

Following the contiguous memory allocation approach, whenever a process has to be granted memory space, we must allot the process a continuous vacant block of space to dwell in. This may be accomplished in two ways:

1. Partitioning Scheme with Fixed Dimensions

2. Partitioning Scheme with Variable Sizes

Let's take a closer look at each of these strategies, as well as their benefits and drawbacks.

Partitioning Scheme with Fixed Dimensions

Each process is given a fixed size continuous block in the main memory in this sort of contiguous memory allocation approach. That is, there will be fixed-size continuous blocks into which the whole memory will be split, and each time a process enters, it will be assigned one of the free blocks. Because each process is given the same amount of memory space regardless of its size. This method is also known as static partitioning.

Advantages

The following are the benefits of a fixed-size partition scheme:

1. This technique is straightforward to implement since all of the blocks are the same size. All that remains is to split memory into fixed chunks and assign processes to them.

2. It is simple to keep track of how many memory blocks remain, which determines how many additional processes may be allocated memory space.

3. Because several processes may be retained in memory at the same time, this approach can be employed in a system that requires multiprogramming[8].

Disdvantages

Although the fixed-size division method offers numerous benefits, it also has several drawbacks:

1. Because the block size is limited, we will not be able to provide space to a process that is larger than the block.

2. The degree of multiprogramming is determined by the size of the blocks, and only as many processes may exist in memory at the same time as the number of blocks.

3. If the block's size is more than the process's size, we have no option but to assign the process to this block, however this will result in a lot of vacant space in the block. This vacant slot may have been utilised for another operation. This is referred to as internal fragmentation. As a result, this strategy may waste space.

Partitioning Scheme with Variable Size

There are no set blocks or divisions in the memory while using this sort of contiguous memory allocation approach. Instead, each process is given a variable-sized block based on its needs. That is, whenever a new process requests memory, this amount of space is allocated to it if it is available. As a result, the size of each block is determined by the size and needs of the operation that fills it.

Advantages

The following are the benefits of a variable-size partition scheme:

1. There is no internal fragmentation since the processes are given blocks of space based on their needs. As a result, there is no memory waste in this approach.

2. The number of processes that may be in memory at the same time is determined by the number of processes in memory and the amount of space they take. As a result, it will vary in various instances and be dynamic.

3. Because there are no fixed-size blocks, even large processes may be given room.

Disadvantages:

Although the variable-size division system offers numerous benefits, it also has several drawbacks:

1. Because this technique is dynamic, implementing a variable-size division system is tricky.

2. It is tough to keep track of processes and available memory space.

3. Paging is a memory management process in which a computer stores and retrieves data from a device's secondary storage to the main storage. Memory management is an essential component of every computer system, and paging in particular is critical to the deployment of virtual memory.

4. There are two kinds of computer memory: main and secondary memory. Data in primary storage is transient and regularly accessed by apps or other hardware components. It is usually kept in random access memory (RAM) for quick retrieval. Secondary storage refers to the storing of data in a computer over extended periods of time. Secondary storage is usually slower than main storage. A solid-state drive (SDD), for example, is an example of secondary memory.

5. Virtual memory, which is what paging is often used for, is a memory management approach in which secondary memory may be accessed as if it were a part of main memory. Paging is a key component of virtual memory since it permits applications in secondary storage to surpass the physical storage's allowed capacity[9].

Working paging

Paging works by writing data to secondary storage and reading it back to main storage. Paging is a fundamental memory management feature for every computer's operating system (OS), including Windows, Unix, Linux, and macOS.

In a memory management system that uses paging, the operating system receives data from secondary storage in blocks called pages, each of which has the same size. A frame is a physical chunk of memory that contains a single page. When paging is employed, a frame in secondary storage does not have to be a single physically continuous section. This strategy has an advantage over previous memory management strategies in that it allows for more efficient and quicker storage use.

Segmentation

Segmentation is a memory management method that divides an application's virtual address space into segments.

The operating system can track which sections of the memory are in use and which are free by dividing the memory into manageable pieces. This makes memory allocation and deallocation significantly quicker and easier for the operating system. The segments are not continuous and are not of equal size. Internal fragmentation does not occur since it is a non-contiguous memory allocation approach. In a user programme, the length is determined by the function of the section.

Segmentation is required.

Because of issues with the paging mechanism, segmentation was created. The paging approach divides a function or piece of code into pages without taking into account that the related pieces of code may also be split. As a result, for the process to run, the CPU must load more than one page into the frames so that the whole associated code is available for execution. Paging required additional pages to load a task into main memory. As a result, segmentation was invented, which divides the code into modules so that similar code may be concatenated in a single block.

Another significant disadvantage of other memory management systems is that the real view of physical memory is isolated from the user's view of physical memory. By splitting the user's software into parts based on the unique requirement, segmentation aids in issue resolution.

Segmentation Types

Segmentation is classified into two categories. They are as follows:

1. Simple Segmentation - With Simple Segmentation, each process is separated into many segments, which are all put into memory at run time. They might be in non-contagious areas.

2. Virtual Memory Segmentation –

In Virtual Memory Segmentation, each process is partitioned into many segments that do not all live in the same location at the same time.

The Benefits of Segmentation

The following are the benefits of the segmentation technique:

The segment table is mostly utilised in the Segmentation method to keep track of segments. Furthermore, the segment table takes significantly less space than the paging table.

There is no internal fragmentation. Segmentation enables us to break the software into modules for easier display. Segments vary in size.

The Drawbacks of Segmentation

The following are some downsides of this technique:

Keeping a segment table for each operation adds overhead.

This procedure is costly.

The time needed to retrieve the instruction rises since two memory visits are now necessary.

Segments in segmentation are of uneven size and hence unsuitable for exchanging.

This strategy causes external fragmentation since the open space is split down into smaller chunks when processes are loaded and deleted from main memory, resulting in a lot of memory waste.

Paging by Segment

Pure segmentation is not widely used and is not supported by many operating systems. However, segmentation and paging may be coupled to gain the greatest benefits of both strategies.

The primary memory is partitioned into variable size segments, which are further divided into fixed size pages in Segmented Paging.

1. Pages are more compact than segments.

2. Because each segment has a page table, each programme has several page tables.

3. The logical address is represented by the Segment Number (base address), the Page Number, and the Page Offset.

It is pointing to the relevant Segment Number.

Page Number It refers to the specific page inside the section.

Working-

The process is first separated into segments, and then each segment is divided into pages in segmented paging.

These pages are then saved in main memory frames.

Each segment has a page table that maintains track of the frames that store the pages of that segment.

Each page table takes up one frame in main memory.

The number of entries in a segment's page table equals the number of pages divided by that segment.

There is a segment table that maintains track of the frames that store the page tables of segments.

The number of entries in a process's segment table equals the number of segments into which the process is split.

The segment table base register stores the segment table's base address.

Operating System 32-Bit

It is a CPU architectural type capable of transferring 32 bits of data. It relates to how much data and information your CPU can quickly handle while running. The vast majority of computers manufactured in the early 2000s and 1990s were 32-bit devices.

Typically, one bit in the register may relate to a single byte. As a result, the 32-bit system can access about 4,294,967,296 bytes (4 GB) of RAM. Its real maximum is less than 3.5 GB (typically) since a section of the register contains temporary data other than memory addresses.

Computer systems can handle information, data, and memory locations represented by 64 bits thanks to the 64-bit microprocessor. A system of this kind may commonly refer to 16 exabytes (17,179,869,184 GB) of memory, or 18,446,744,073,709,551,616 bytes.

More than 4 GB of RAM may be accessed by a 64-bit system (a computer with a 64-bit CPU). It is several million times more than what a typical workstation would need to access. It indicates that a machine with 8 GB of RAM needs a 64-bit CPU. Otherwise, the CPU will be unable to access at least 4 GB of memory[10].

Benefits of a 32-bit processor

Here are some of the most major advantages and disadvantages of a 32-bit processor: The sole benefit that can be mentioned is that it is compatible with all older devices that were produced between the early 2000s and the late 1990s.

The Benefits of a 64-Bit Processor

Here are some advantages and disadvantages of utilising a 64-bit processor:Improved application performance and the ability to use a 64-bit operating system. Improved security feature. Using Windows 64-bit with a recent 64-bit CPU enables you to take use of extra security features that 32-bit users do not have.A 64-bit CPU provides security measures that go beyond hardware kernel patch protection.A 64-bit CPU enables the creation of 16TB of virtual memory. Although 8 TB is reserved for user processes and 8 TB for kernel processes.64-bit CPUs provide enhanced functionality. It has a capacity of 264 computational values.A 64-bit computer can support up to 16.8 terabytes of RAM.64-bit processors are available in dual-core, six-core, quad-core, and eight-core configurations.

Support for many cores allows you to execute more computations, which increases processing power and allows your computer to operate quicker.Software applications that need different sorts of may run effectively on multi-core 64-bit CPUs.Provides access to virtual memory per process.

The Drawbacks of a 32-Bit Processor

The following are significant disadvantages of utilizing a 32-bit operating system:The most significant disadvantage is that companies no longer produce software for 32-Bit operating systems.A 64-Bit operating system is required by many CPUs.Due to a lack of market demand or product, manufacturers often do not provide 32-Bit driver versions for their hardware.

The Drawbacks of a 64-Bit Processor

Here are several disadvantages of utilizing a 64-bit processor:64-bit drivers for older systems and hardware are very unlikely to be released.Some older 32-bit applications do not transfer cleanly to 64-bit.

REFERENCES

[1] A. Pupykina and G. Agosta, "Survey of Memory Management Techniques for HPC and Cloud Computing," IEEE Access. 2019. doi: 10.1109/ACCESS.2019.2954169.

[2] D. Mishra and P. Kulkarni, "A survey of memory management techniques in virtualized systems," Comput. Sci. Rev., vol. 29, pp. 56–73, Aug. 2018, doi: 10.1016/j.cosrev.2018.06.002.

[3] X. Peng et al., "Capuchin: Tensor-based GPU memory management for deep learning," in International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS, 2020. doi: 10.1145/3373376.3378505.

[4] D. Raghuvanshi, "Memory Management in Operating System," Int. J. Trend Sci. Res. Dev., 2018, doi: 10.31142/ijtsrd18342.

[5] Z. Zhu, F. Wu, J. Cao, X. Li, and G. Jia, "A Thread-Oriented Memory Resource Management Framework for Mobile Edge Computing," IEEE Access, 2019, doi: 10.1109/ACCESS.2019.2909642.

[6] M. Parkinson et al., "Project snowflake: Non-blocking safe manual memory management in .NET," Proc. ACM Program. Lang., 2017, doi: 10.1145/3141879.

[7] L. Wang et al., "SuperNeurons: Dynamic GPU Memory Management for Training Deep Neural Networks," ACM SIGPLAN Not., 2018, doi: 10.1145/3178487.3178491.

[8] J. Y. Hur, "Contiguity Representation in Page Table for Memory Management Units," IEEE Trans. Very Large Scale Integr. Syst., 2019, doi: 10.1109/TVLSI.2018.2870913.

[9] F. Balasa, N. Abuaesh, C. V. Gingu, I. I. Luican, and H. Zhu, "Energy-aware memory management for embedded multidimensional signal processing applications," Eurasip J. Embed. Syst., 2017, doi: 10.1186/s13639-016-0043-9.

[10] L. Liu, Y. Li, C. Ding, H. Yang, and C. Wu, "Rethinking Memory Management in Modern Operating System: Horizontal, Vertical or Random?," IEEE Trans. Comput., 2016, doi: 10.1109/TC.2015.2462813.

CHAPTER 11

VIRTUAL MEMORY

Vishal Sharma, Assistant Professor
Department of SOMFT, IIMT University, Meerut Uttar Pradesh, India
Email id- journalistdrvishal@gmail.com

Virtual memory is a memory management approach that allows secondary memory to be utilised as if it were primary memory. Virtual memory is a method that is often utilised in computer operating systems (OS).Virtual memory enables a computer to compensate for physical memory shortages by temporarily shifting data from random access memory (RAM) to disc storage. Mapping memory chunks to disc files allows a computer to utilise secondary memory as if it were main memory.

Most personal computers (PCs) nowadays have at least 8 GB (gigabytes) of RAM. However, this is not always enough to execute many applications at the same time. This is where virtual memory enters the picture. Virtual memory frees up RAM by transferring data that hasn't been utilised in a while to a storage device like a hard disc or solid-state drive (SSD).

Virtual memory is essential for increasing system speed, multitasking, and running huge applications. However, users should not depend too much on virtual memory since it is significantly slower than RAM. If the operating system needs to swap data between virtual memory and RAM too often, the machine will begin to slow down, a process known as thrashing.

Virtual memory was created at a period when actual memory, commonly known as RAM, was prohibitively costly. Because computers have a limited quantity of RAM, memory will ultimately run out if numerous applications are running at the same time. A virtual memory system emulates RAM by utilising a part of the hard disc. A system may use virtual memory to load bigger or numerous applications at the same time, allowing each one to function as if it had more capacity without needing to buy extra RAM[1].

Functioning virtual memory

Virtual memory operates using both hardware and software. When an application is running, data from that programme is saved in RAM at a physical address. A memory management unit (MMU) transfers the address to RAM and translates addresses automatically. The MMU, for example, may link a logical address space to a physical address.

If the RAM space is ever required for anything more important, data may be moved out of RAM and into virtual memory. The memory manager on the computer is in charge of keeping track of the transitions between physical and virtual memory. If the data is required again, the computer's MMU will continue execution through a context switch.

The OS splits memory with a defined number of addresses into pagefiles or swap files when moving virtual memory into physical memory. Each page is kept on a disc, and when a page is required, the operating system transfers it from the disc to main memory and converts virtual addresses to actual addresses.

However, the process of switching from virtual to physical memory is fairly sluggish. This implies that employing virtual memory typically results in a considerable decrease in performance. Because of swapping, computers with more RAM are thought to run better[2].

Virtual memory types

The MMU of a computer controls virtual memory operations. The MMU hardware is often built into the central processor unit of modern computers (CPU). The virtual address space is also generated by the CPU. Virtual memory is often paged or split.

Memory is divided into portions or paging files via paging. When a computer's available RAM is depleted, pages that are not in use are moved to the hard drive through a swap file. A swap file is a space on the hard drive reserved for usage as a virtual memory extension for the computer's RAM. When the swap file is required, it is returned to RAM through a process known as page swapping. This method guarantees that the operating system and apps on the computer do not run out of actual memory. The page file may have a maximum size of 12 to four times the computer's actual memory.

Page tables are used in the virtual memory paging process to transform the virtual addresses used by the OS and programmes into the physical addresses used by the MMU. The page table entries show whether or not the page is in RAM. If the operating system or a programme cannot locate what it requires in RAM, the MMU replies to the missing memory reference with a page fault exception, instructing the OS to transfer the page back to memory when it is required. When a page is loaded into RAM, its virtual address is recorded in the page table.

Virtual memory is also managed via segmentation. This method splits virtual memory into parts of varying lengths. Memory segments that are no longer in use may be relocated to virtual memory space on the hard disc. A segment table keeps track of whether or not a segment is existing in memory, if it has been updated, and what its physical address is. Furthermore, file systems in segmentation are only made up of segments that are mapped into the possible address space of a process.

In terms of how memory is partitioned, segmentation and paging vary as memory models; nonetheless, the operations may be coupled. Memory is organised into frames or pages in this situation. The segments occupy many pages, and the virtual address contains both the segment number and the page number.

Other techniques of page replacement include first-in-first-out (FIFO), optimum algorithm, and least recently used (LRU). Memory selects the replacement for a page that has been in the virtual address for the longest period using the FIFO approach. The best algorithm technique chooses page replacements depending on which page is most likely to be replaced after the longest period of time; although complex to implement, this results in fewer page faults. The LRU page replacement technique replaces the page in main memory that has not been utilised for the longest period[3].

Virtual memory management

Managing virtual memory inside an operating system is simple since there are default settings that define how much hard disc space to assign for virtual memory. These settings will work for most apps and processes, but there may be instances when you need to manually reset the amount of hard drive space allotted to virtual memory, such as when using applications that need rapid reaction times or when the computer has several hard disc drives (HDDs).

The minimum and maximum amount of hard disc space to be utilised for virtual memory must be specified when manually resetting virtual memory. Allocating insufficient HDD space for virtual memory might cause a computer to run out of RAM. If a system often requires extra virtual memory space, it may be prudent to explore RAM expansion. Common operating systems may typically advise users not to expand virtual memory over 12 times the amount of RAM.

The way virtual memory is managed varies depending on the operating system. As a result, IT workers must grasp the fundamentals of managing physical memory, virtual memory, and virtual addresses[4].

RAM cells in SSDs have a finite lifetime as well. Because RAM cells have a finite amount of writes, utilising them for virtual memory often lowers the drive's lifetime.

Advantages of utilising virtual memory:

Virtual memory has the following advantages:

It can handle twice as many addresses as main memory.

It allows for the usage of many apps at the same time.

It relieves programmes of the burden of maintaining shared memory and eliminates the need for users to add memory modules when RAM space runs out.

It has improved performance when just a portion of a programme is required for execution.

It offers improved security due to memory separation.

It allows numerous bigger programmes to operate concurrently.

Allocating RAM is a low-cost operation.

No external fragmentation is required.

Using the CPU to manage logical partition workloads is beneficial.

Data may be sent automatically.

During a fork system call action that generates a clone of itself, pages in the original process may be shared.

In addition to these advantages, managers in a virtualized computing environment may employ virtual memory management methods to assign more memory to a virtual machine (VM) that has exhausted its resources. These virtualization management techniques help boost VM performance and administration flexibility.

The drawbacks of utilising virtual memory

Although the usage of virtual memory provides advantages, there are certain drawbacks to consider, such as:

i. Applications perform slower when they use virtual memory.

ii. Data must be translated between virtual and physical memory, which necessitates additional hardware support for address translations, thus slowing down a computer.
iii. The quantity of secondary storage as well as the computer system's addressing technique restrict the capacity of virtual storage.
iv. Thrashing may occur if there is insufficient RAM, causing the machine to run slowly.
v. It may take longer to transition between apps when utilising virtual memory, and it reduces available hard drive space[4].

Physical memory vs. virtual memory (virtual RAM) (RAM)

When discussing the distinctions between virtual and physical memory, the most prominent contrast is speed. RAM is much quicker than virtual memory. RAM, on the other hand, is more costly.

When a computer needs storage, RAM is the first thing that is utilised. When the RAM is full, virtual memory, which is slower, is utilised. This graph compares virtual RAM (virtual memory) to RAM (physical memory).

Users may actively increase RAM to their computers by purchasing and installing additional RAM chips. This is important if they are suffering slowdowns as a result of too many memory swaps. The quantity of RAM on a computer is determined by what is installed. The capacity of the computer's hard drive, on the other hand, limits virtual memory. Virtual memory settings are often regulated by the operating system.

Furthermore, RAM employs swapping mechanisms, while virtual memory employs paging. Virtual memory is restricted by the capacity of the hard drive, while physical memory is limited by the size of the RAM chip. RAM has direct access to the CPU as well, but virtual RAM does not.

OS Demand Paging:

According to the Virtual Memory idea, in order to operate a process, only a portion of the process has to be present in main memory, which implies that only a few pages will be stored in main memory at any one moment.

However, selecting which pages should be retained in main memory and which should be stored in secondary memory will be challenging since we cannot predict when a process will need a certain page.

As a result, a concept known as Demand Paging is established to address this issue. It advises storing all frame pages in secondary memory until they are needed. In other words, it indicates that no page should be loaded into main memory unless it is needed.

When a page is referred to for the first time in the main memory, it is discovered in the secondary memory[5].

Methods of Page Replacement

Here are some useful Page replacement solutions.

• FIFO

Algorithm Optimization

LRU Page Replacement

FIFO Page Substitution

First-in-first-out (FIFO) is a straightforward implementation approach. Memory chooses the page for a replacement that has been in the virtual address of the memory for the longest duration in this manner.

• When a new page is loaded, the most recently loaded page is deleted from memory. As a result, determining which page has to be destroyed is simple since its identification number is always at the top of the FIFO stack.

• The oldest page in the main memory should be chosen for replacement first.

The Best Algorithm

The optimum page replacement technique chooses the page for which the time to the next reference is the longest.

• The optimal method produces the fewest number of page faults. This algorithm is tough to put into practise.

• Of all methods, an optimum page-replacement algorithm technique has the lowest page-fault rate. This algorithm exists and should be known as MIN or OPT.

• Replace the page that you don't want to utilise for a long time. It only consumes the time when a page is required[6].

OS Demand Paging Implementation

The following are the primary stages involved in demand paging between when a page is requested and when it is loaded into main memory:

1. CPU refers to the page that is required.

2. The referenced page is examined in the page table to see whether it is present in main memory.

Otherwise, an interrupt page fault is triggered.

3. The operating system places the interrupts process in a blocking state and begins the process of retrieving the page from memory so that the process can be executed.

4. The operating system will look for it in logical address space.

5. The required page will be transferred from logical to physical address space. Page replacement algorithms are used to make decisions about replacing pages in physical address space.

6. The table on the page will be updated.

7. The interrupted procedure will be resumed.

Common Terms in Demand Paging Operating System

The following are some terminology used in demand paging operating systems:

• Page Error • Swapping

• Thrashing Page Error

If the referred page is not present in main memory, there will be a miss; this is known as a page miss or page fault.

The CPU must search secondary memory for the lost page. When the number of page faults is high, the effective access time of the system rises considerably.

Swapping

Swapping entails either wiping all of the process's memory pages or marking them so that we may delete them using the page replacement approach.

A suspended process signifies that it is unable to run. However, we may temporarily alter the procedure. Over time, the system might switch the process from secondary memory to main memory. Thrashing is a circumstance in which a process is active and pages are switched in and out of it[7].

Thrashing

If the number of page faults is equal to the number of referred pages or if the number of page faults is so large that the CPU is only reading pages from the secondary memory, the effective access time will be the time required by the CPU to read one word from the secondary memory. This is referred to as thrashing.

The Benefits of Demand Paging

The following are the advantages of employing the Demand Paging technique:

• Memory is used effectively with the aid of Demand Paging.

• Demand paging eliminates the need for external fragmentation.

• Demand paging requires less input/output.

• The amount of physical memory has no bearing on this operation.

• Demand Paging makes it easy to distribute pages.

• Parts of the process that are never called are never loaded using this method.

• There is no need for compaction in demand paging.

The Drawbacks of Demand Paging

The following are the disadvantages of Demand Paging:

• Overheads have increased owing to interruptions and page tables.

• Memory access time is longer in demand paging.

Operating System Page Replacement

In operating systems that employ virtual memory with Demand Paging, page replacement is required. As we know, demand paging loads just a subset of a process's pages into memory. This is done so that multiple processes may run concurrently in memory.

When a process requests a page from virtual memory for execution, the Operating System must determine which page will be replaced by this requested page. This is known as page replacement, and it is an important part of virtual memory management[8].

Page Replacement Algorithms are required

To understand why we need page replacement techniques, we must first understand page faults. Let's look at what a page fault is.

Page Error: A Page Fault happens when a programme executing in the CPU attempts to access a page in the program's address space, but the requested page is presently not loaded into the system's primary physical memory, the RAM.

Page errors arise because the real RAM is substantially smaller than the virtual memory. As a result, if a page fault occurs, the operating system must replace a previously requested page in RAM with the newly requested page. Page replacement algorithms aid the Operating System in determining which page to replace in this case. The main goal of all page replacement algorithms is to reduce the amount of page defects.

Page replacement types

FIFO Algorithm for Page Replacement

It is a fairly basic method of Page replacement known as First in First Out. This approach primarily replaces the oldest page in main memory that has been there for the longest period.

OS File Allocation

When a hard disc is formatted, a system has multiple small places called blocks to hold data. The operating system stores data in memory blocks using different file allocation techniques, allowing the hard drive to be utilised more effectively and files to be accessed.

File Allocation Methods in Operating Systems • Contiguous allocation method • Linked allocation method • Indexed allocation method • File Allocation Table (FAT) • Inode allocation method

Managing Kernel Memory

Managing Kernel Memory When a process operating in user mode asks more memory, pages are allocated from the kernel's list of free page frames. Also, keep in mind that if a user process asks a single byte of memory, internal fragmentation will occur since the process will be given a full page frame. Kernel memory, on the other hand, is often allocated from a separate free-memory pool than the one used to satisfy regular user-mode activities. This is due to two fundamental reasons: 1. The kernel requires memory for various data structures, some of which are smaller than a page in size. As a consequence, the kernel must utilise memory sparingly and strive to reduce waste caused by fragmentation. This is critical since many operating systems do not expose kernel code or data to the paging system. 2. Pages assigned to user-mode processes do not have to be physically contiguous. Certain hardware devices, on the other hand, communicate directly with physical memory without the advantage of a virtual memory interface and, as a result, may need memory to be stored in physically continuous pages. In the sections that follow, we will look at two ways for maintaining free memory given to kernel processes[9], [10].

Operating System Examples

1. Windows from Microsoft. Without a doubt the most popular of the operating systems, but it is really a collection of distributions (an operating environment) designed to offer earlier operating systems (such as MS-DOS) with a graphical interface and a set of software tools. Its initial version was released in 1985. Since then, it has not ceased upgrading in increasingly powerful and diversified versions, as Microsoft, its parent company, dominates the digital technology industry.

2 GNU / Linux is the second option. This phrase refers to the usage of a free Unix kernel named "Linux" in conjunction with the free GNU distribution. As a consequence, one of the key players in the creation of free software, whose source code may be freely used, changed, and redistributed, has emerged.

3. UNIX. This portable, multi-tasking, multi-user operating system was created in early 1969, and its copyright rights have shifted from one business to another over the years. In truth, it is a family of comparable operating systems, many of which have become commercial and others are free, all based on the Linux kernel.

4. Fedora. It is basically a general-purpose Linux distribution that arose after the demise of Red Hat Linux, with whom he is closely associated, but which arose as a community initiative. It is another word that comes to mind when discussing free software and open source in its three primary forms: workstation, cloud, and server.

5. Ubuntu. This free and open-source operating system is based on GNU / Linux and derives its name from a South African philosophy centred on man's devotion to the rest of the species. In this way, Ubuntu is aimed toward simplicity and freedom of use, despite the fact that Canonical, the British business that holds its rights, makes its living via technical services related to the software.

6. MacOS. The Macintosh operating system, usually known as OSX or Mac OS X, is a Unix-based environment that has been developed and marketed as part of Apple computers since 2002. Apple introduced Darwin, an open and free source operating system, to which they eventually added components such as Aqua and the Finder to achieve the interface on which Mac OS X, its most current version, is based.

REFERENCES

[1] E. Krokos, C. Plaisant, and A. Varshney, "Virtual memory palaces: immersion aids recall," Virtual Real., 2019, doi: 10.1007/s10055-018-0346-3.

[2] T. Hussain and K. M. Quinn, "Similar but different: virtual memory CD8 T cells as a memory-like cell population," Immunology and Cell Biology. 2019. doi: 10.1111/imcb.12277.

[3] A. Bhattacharjee and D. Lustig, “Architectural and Operating System Support for Virtual Memory,” Synth. Lect. Comput. Archit., 2017, doi: 10.2200/s00795ed1v01y201708cac042.

[4] J. Gandhi et al., “Range Translations for Fast Virtual Memory,” IEEE Micro, 2016, doi: 10.1109/MM.2016.10.

[5] W. A. Bhat, A. Rashid, F. F. Wani, and F. Altaf, “Virtualization and visualization of virtual memory system for effective teaching–learning,” Comput. Appl. Eng. Educ., 2019, doi: 10.1002/cae.22152.

[6] B. Suchy, S. Campanoni, N. Hardavellas, and P. Dinda, “CARAT: A case for virtual memory through compiler- and runtime-based address translation,” in Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI), 2020. doi: 10.1145/3385412.3385987.

[7] P. Vogel, A. Marongiu, and L. Benini, “Exploring Shared Virtual Memory for FPGA Accelerators with a Configurable IOMMU,” IEEE Trans. Comput., 2019, doi: 10.1109/TC.2018.2879080.

[8] M. Pribikova, A. Moudra, and O. Stepanek, “Opinion: Virtual memory CD8 T cells and lymphopenia-induced memory CD8 T cells represent a single subset: Homeostatic memory T cells,” Immunology Letters. 2018. doi: 10.1016/j.imlet.2018.09.003.

[9] S. K. Peddoju, H. Upadhyay, J. Soni, and N. Prabakar, “Natural language processing based anomalous system call sequences detection with virtual memory introspection,” Int. J. Adv. Comput. Sci. Appl., 2020, doi: 10.14569/IJACSA.2020.0110559.

[10] D. YA, “An Efficient Virtual Memory using Graceful Code,” Int. J. Trend Sci. Res. Dev., 2019, doi: 10.31142/ijtsrd23878.

CHAPTER 12

MASS STORAGE SYSTEM

DR. VIKAS SHARMA, ASSISTANT PROFESSOR
Department of Computer Science Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id-vikass.soeit@sanskriti.edu.in

A mass storage device (MSD) is any storage device that allows for the storing and transfer of huge volumes of data among computers, servers, and within an IT environment. MSDs are portable storage devices that offer a storage interface that may be both internal and external to the computer. A mass storage device may also be referred to as an auxiliary storage device. The phrase is often used to denote USB mass storage devices.

Structure Disk

The smallest transfer unit used by modern disc drives is the logical block, which is treated as a big one-dimensional array. Although certain drives may be low-level formatted to have a variable logical block size, such 1,024 bytes, a logical block is typically 512 bytes in size. The sectors of the disc are successively mapped into the one-dimensional array of logical blocks. On the furthest cylinder, Sector 0 is the first sector of the first track. The mapping moves from outermost to innermost cylinders starting with that track, then through the other tracks in that cylinder, and finally through the remaining cylinder. May translate a logical block number into an old-fashioned disc address by utilising this mapping, which consists of a cylinder number, a track number within that cylinder, and a sector number inside that track. This translation is challenging to execute in practise for two reasons.

A. First, the majority of discs include a few bad sectors, but the mapping masks this by using extra sectors from other parts of the disc.
B. Second, on certain drives, the number of sectors per track varies.

The bit density per track on media using constant linear velocity (CLV) is constant. A track's length increases with distance from the disk's centre, increasing the number of sectors it can store. The number of sectors per track reduces from outer zones to inner zones. Typically, tracks in the outer zone may contain 40% more sectors than tracks in the inner zone. As the head transitions from the outer to the inner tracks, the drive speeds up its rotation to maintain the same rate of data transfer underneath the head (CD-ROM, DVD-ROM drives)[1].

Disk Scheduling: Since several operations may be carried out simultaneously on a single computer, it is important to manage all processes that are currently active on the system. We can run many programs at once with the aid of multi-programming. So the operating system employs the concept of disc scheduling to control and provide memory to all processes.

This divides the CPU's time amongst the numerous processes and establishes whether or not each one will function properly. As a result, disc scheduling will specify when the CPU will run each process. Therefore, scheduling entails carrying out all of the tasks that are assigned to a CPU at once.

Scheduling Strategies

To allow many processes to run simultaneously at once, the CPU's total processing time is divided among them using scheduling. The CPU employs the following scheduling techniques to divide or share the total time available to it.

1) FCFC, or first come, first served, refers to the execution of tasks or processes in the same order as they are input into the computer. Create a queue in this operating system that includes the order in which each task is to be completed and the order in which the CPU will carry out each task. All jobs in this are carried out in the sequence in which they were entered. The CPU will begin working on the Job that was requested initially in this scenario. Additionally, jobs that are entered later will be executed in the order of entry.

1) SSTF, or Shortest Seek Time First: In this technique, the Operating System looks for the task that requires the least amount of CPU time to run in order to find the shortest time possible. Following a thorough examination of each work, the jobs are either organised into priority order or sequence. The total amount of time a process uses to execute will depend on its priority. The shortest seek time will include total seek time, that is, seek time to enter and seek time to process completion. is the whole amount of time a process will take to complete.

4) C-Scan Planning: All of the processes in the C-Scan are organised using a circular list. A circular list is one that has no beginning or ending points, which implies that the list's end serves as its first starting point. The CPU will look for the process in the C-Scan Scheduling from beginning to finish, and if an end has been found, the search will restart at the beginning. Because a CPU often runs many processes at once, a user may wish to enter some data at that time. In such case, the CPU will restart the running process after the input operation. C-Scan Scheduling will then be utilised to process the same processes repeatedly.

5) Look Scheduling: During the Look Scheduling, the CPU scans the list of processes that are running on the disc from beginning to finish. It also scans the whole disc from one end to the other.

1) Round Robin. The Round Robin Scheduling, also known as Quantum Time, divides the CPU's time into equal numbers. Each process that is a request for execution uses the CPU an equal number of times, and when the first process has finished, the CPU moves on to the next one automatically. But the main issue is that even once the process is over, time will still be consumed by it. That is to say, whether a process has any operations to do or not, the CPU will still use up time, which is a waste of CPU time.

2) Priority Scheduling: In this method, each process has a set of priorities assigned to it. This means that each process will be evaluated based on the total time it will take to complete. Setting the Priorities of the Processes will include looking at the Total Time of the Process, the Number of Times a Process Needs Some Input and Output, and the Number of Resources. In order for the CPU to process all of the processes, they must first be organised into the form of these criteria.

Multilevel Queue: As we are aware, there are several different sorts of tasks that must be carried out on computers simultaneously. The Multilevel Queue is utilised when there are multiple queues for the various distinct processes. The CPU uses this technique to maintain the queues in order to organise the many or distinct types of queues. In this, all of the queues are gathered and organised in the shape of various tasks that people are seeking to be completed. In order to maintain the many types of queues, which include all the processes with the same kind[2], [3].

Managing discs in an operating system

Modern operating systems provide a wide variety of services and add-ons, and all operating systems execute the same four fundamental operating system administration tasks. The next section provides a short description of these management tasks together with the following broad background. The four primary operating system administration tasks—each of which is covered in further detail elsewhere are:

A. Process Control
B. Memory administration
C. Management of files and discs
D. I/O System Administration

The majority of computer systems use secondary storage components (magnetic disks). It offers inexpensive, non-volatile storage for applications and data (tape, optical media, flash drives, etc.). Programs are housed on unique storage units called files, as are the user data they utilise. The operating system is in charge of reserving space on secondary storage devices as required for files. Large files, in particular, cannot be guaranteed to be stored on physical disc drives in a contiguous manner. The quantity of space at hand has a big role. New files are more likely to be recorded in numerous places when the disc is full. The operating system's sample file, however, conceals the fact that the file is divided into many pieces from the user from view. Every component of each file on the disc has to be tracked by the operating system in terms of where it is on the disc. It may be necessary to keep track of tens of thousands of files and file fragments on a single physical drive in certain situations. The operating system must furthermore be able to find each file and execute read and write operations on it as necessary. Because of this, the operating system is in charge of setting up the file system, assuring the security and dependability of reading and writing activities to secondary storage, and maintaining access times (the time required to write data to or read data from secondary storage). Tools for managing discsDisk management programmes assist in finding and fixing drive issues, setting up a disc for data storage, and deleting unneeded files[4], [5].

Software for managing discs comprises the following:

Fdisk is a command-line programme that may be used to create and remove hard drive partitions as well as designate active or boot partitions.

Formal: Used to set up a hard disc for data storage.

By scanning the disc surface for physical faults, Scandisk or Chkdsk is used to verify the integrity of the data and folders on a hard drive.

Defrag: Used to maximise hard disc space so that applications and data may be accessed more quickly.

Disk Cleanup is a programme that looks for items that may be securely erased in order to free up space on a hard disc.

Disk management: Used to start discs, create partitions, and format partitions; manage hard drives and partitions.

For file troubleshooting and repair, use the Windows XP boot disc. The Windows XP boot CD can fix Windows system data, recover deleted or damaged files, and reinstall Windows. There are software tools made by third parties that may help in problem-solving.

Swap-Space Management

Another low-level operation performed by the operating system is swap-space management. Disk space is used by virtual memory to supplement main memory. Using swap space severely lowers system performance since disc access is considerably slower than memory access. The optimum throughput for the virtual memory system is the primary objective for the design and implementation of swap space. Depending on the memory-management algorithms in use, different operating systems use swap space in unique ways. Systems that employ switching, for instance, may store a whole process image, including the code and data segments, in swap space. Pages that have been shifted out of the main memory may just be stored by paging systems. Depending on the quantity of physical memory, how much virtual memory it is supporting, and how it is utilised, a system may need a different amount of swap space. It may be anything between a few megabytes and a gigabit of disc space. It should be noted that it may be better to estimate the quantity of swap space needed too high than too low since if a system runs out of swap space, it may be compelled to kill processes or crash completely. Overestimation is harmless, although it consumes disc space that may be utilised for files. The amount to reserve for swap space is advised by certain systems. For instance, Solaris advises setting swap space to the difference between virtual memory and physically pageable memory. Although most Linux systems today utilise far less swap space, Linux formerly advised configuring swap space to twice the amount of physical memory. There is now a heated discussion in the Linux community regarding whether or not swap space should even be reserved.Linux is one of the operating systems that supports numerous swap areas. In order to distribute the strain exerted on the I/O system by paging and swapping among the system's I/O devices, these swap spaces are often placed on different discs[6], [7].

Swap Space's Uses

There are several ways that Swap-space is used by various operating systems. It is possible for the systems using swapping to store the full process, including image, code, and data segments, in swap space. As many processes share the CPU, swapping is a memory management strategy used in multi-programming. This method involves taking a process out of the main memory, putting it in secondary memory, and then bringing it back for further execution. Swap Out refers to the process of transferring a process from primary memory to secondary memory. Swapping In refers to the process of shifting a process from secondary memory to main memory.

Pages that have been shifted out of the main memory may just be stored by paging systems. Megabytes to gigabytes of swap space may be required on a machine. However, it also relies on how much virtual memory it is supporting, how much physical memory it has, and how it utilises that virtual memory.

The file system is the component of any operating system that most users can immediately identify. This gives users a way to store and retrieve data, as well as operating system applications that are accessible to all computer system users. There are two separate components to the file system:

A. A directory structure that organises and offers details about all the files in the system, along with a group of files that hold relevant data.
B. The many file formats, file concepts, file storage, and file operations in this chapter.

A file has a name so that its users can refer to it easily. A name typically consists of a string of characters, such as filename.cpp, with an extension that indicates the file type. Other systems do not differentiate between capital and lowercase letters in names, unlike certain systems (like Linux). A file becomes independent of the process, the user, and the system that generated it when it is given a name. Let's assume that one user created the file filename.cpp and that another user is modifying it by inferring its name. The owner of the file may transfer it over a network, transmit it through email, or write it on a compact disc (CD), but the destination system may still refer to it as filename.cpp. The properties of a file vary from operating system to operating system but commonly include these:

Name: The symbolic file name is the sole piece of information that is preserved in a readable manner.

In the non-human readable version of the file, the identifier is a number that uniquely identifies the file inside the file system.

Type: Systems that accept various file types or their formats need this information.

Location: This data is a device pointer pointing to the file's location on the device where it is stored.

Size: This characteristic includes the file's current size (measured in bytes, words, etc.) and, if applicable, the file's maximum permitted size.

Protection: Information on access-control determines who is permitted to read, write, execute, etc.

Date, time, and user identification: This data may be stored for the file's creation, last change, and most recent usage. In the areas of protection, security, and use tracking, this data may be helpful.

An abstract data type is a file. The actions that may be carried out on files must be taken into account when correctly defining a file. To create, write, read, move, delete, and truncate files, the operating system may provide system calls. An operating system has six fundamental file operations. Which are:

The two stages involved in generating a file are as follows. The file must first be able to fit anywhere in the file system. We talk about how to set aside space for the file. Second, a directory entry for the new file has to be created[8], [9].

Create a file: You must supply both the name of the file and the data to be written to it in a system call in order to write to a file.

Examining a file: When reading from a file, you use a system call that provides the file's name and where the next block of the file should be stored in memory.

Rearranging a file's contents: The "current-file-position" pointer is then relocated to the specified value once the directory has been searched for the appropriate item. Moving inside a file doesn't always need to involve real I/O. File seek is another name for this file activity.

Delete a file: To delete a file, look for the particular file in the directory. By deleting the file or directory, all available file space is released for usage by other files.

Truncating a file: The user may want to delete the file's content while maintaining its properties. This programme enables all characteristics to stay unaltered, with the exception of file length, and lets the user add or update the file content rather than destroying the file and then recreating it.

File Concepts:

Computers use storage devices like discs, tape drives, and optical discs to store information in files. The operating system offers a logical representation of the data on the disc. A file is this logical storage unit. Since the data in files is non-volatile, it is not lost when the power goes off. A file is referred to as a collection of connected data that is physically kept. Data must first be written to a file in order to be written to a computer. In general, a file is a collection of bits, bytes, lines, or records that its owner or author has specified. Depending on the file type, a file's structure is established by its owner or author.

Text file: It is made up of a string of characters.

Image files include visual data like pictures, vector graphics, and more.

The source file contains functions and subroutines that are subsequently compiled.

Object file: The linker uses this file, which contains a series of bytes divided into bytes.

The binary code that the loader loads into memory and executes is kept in an executable file (.exe).

Methods of File Access in the Operating System

There are numerous methods to access the information in a file when it is utilised, and information from the file is read and accessible into computer memory. Some systems only provide one way to access files. Other systems, like those of IBM, enable a variety of access mechanisms, thus deciding which one is best for a given application is a significant design challenge. A file may be accessed using one of three methods: sequential access, direct access, or the index sequential method.

The simplest access technique is sequential access. One record at a time, the information in the file is processed in sequence. This method of accessing the file is by far the most used; for instance, editors and compilers often do so.

The majority of file operations are read and write. A file pointer that tracks the location of I/O operations is automatically advanced during a read operation called "read next" that reads the file's next position. In a similar manner, advance to the newly written information with the -write next- command and add to the end of the file. Data is accessed sequentially, one record after another. When we perform the read command, the cursor advances by one. When we execute the write command, memory will be allocated, and the pointer will be moved to the end of the file. This approach makes sense for tape.

Sequential access method benefits:

A. This file access technique is easy to build.
B. To get the next item rapidly, it employs lexicographic order.

Sequential Access Method drawbacks:

This kind of file access technique is sluggish if the file record that has to be accessed next is not present next to the current record. To insert a new record, a large portion of the file may need to be moved.

Direct Access - A second technique is direct access, sometimes called relative access. a logical record with a predetermined length that the software may read and write quickly, in any sequence. Since disc enables random access to any file block, the direct access is based on the disc model of a file. The file is seen as a numbered series of blocks or records for immediate access. As a result, we may read block 14 and block 59 before writing block 17 itself. The reading and writing sequence for a direct access file is not constrained. A relative block number is one that the user typically provides to the operating system; the first relative block of a file is 0; the second is 1; and so on.

Direct Access Method Benefits:

The files may be accessible quickly, reducing the typical access time.

The direct access approach eliminates the requirement to traverse all the blocks that come before a block in order to reach it.

Index sequential technique: This alternative approach to file access is based on the sequential access method. These techniques create the file's index. The pointer to each block is included in the index, much like an index at the back of a book. We first search the index to discover a record in the file, and then we use the pointer to access the file directly[10].

REFERENCES

[1] F. G. Battisti, G. S. Delsoto, and A. K. da Silva, “Transient analysis and optimization of a recuperative sCO2 Brayton cycle assisted by heat and mass storage systems,” Energy, 2018, doi: 10.1016/j.energy.2018.02.045.

[2] T. Nakamura, K. Nagano, and S. Kindaichi, “A study on the energy evaluation of the building thermal mass storage system with the chilled water storage system,” J. Environ. Eng., 2010, doi: 10.3130/aije.75.289.

[3] L. Vickovic, S. Gotovac, and S. Celar, “Simulation-based performance analysis of the ALICE mass storage system,” Int. J. Simul. Model., 2016, doi: 10.2507/IJSIMM15(1)6.325.

[4] M. Choi, N. Park, V. Piuri, and F. Lombardi, “Reliability measurement of mass storage system for onboard instrumentation,” IEEE Trans. Instrum. Meas., 2005, doi: 10.1109/TIM.2005.858514.

[5] S. Bokhari, B. Rutt, P. Wyckoff, and P. Buerger, “Experimental analysis of a mass storage system,” Concurr. Comput. Pract. Exp., 2006, doi: 10.1002/cpe.1038.

[6] H. Jin and L. Ran, “A Fair-Rank Ant Colony Algorithm in Distributed Mass Storage System,” Can. J. Electr. Comput. Eng., 2015, doi: 10.1109/CJECE.2015.2469597.

[7] X. Li, T. Cai, and S. Ju, “The hash postfix table based metadata management algorithm for mass storage system,” Int. J. Digit. Content Technol. its Appl., 2010, doi: 10.4156/jdcta.vol4.issue9.12.

[8] H. Muramatsu, E. Togashi, and T. Nobe, “Energy performance evaluation of thermo-active building system based on simulation and measurement,” J. Environ. Eng., 2019, doi: 10.3130/aije.84.759.

[9] Y. Chen et al., “Experimental investigation of demand response potential of buildings: Combined passive thermal mass and active storage,” Appl. Energy, 2020, doi: 10.1016/j.apenergy.2020.115956.

[10] Q. Le, A. Amer, and J. A. Holliday, “RAID 4SMR: RAID Array with Shingled Magnetic Recording Disk for Mass Storage Systems,” J. Comput. Sci. Technol., 2019, doi: 10.1007/s11390-019-1946-4.

CHAPTER 13

DIRECTORY STRUCTURE

Dr. Rajbhadur Singh, Assistant Professor
Department of Computer Science Engineering, Sanskriti University, Mathura, Uttar Pradesh, India
Email Id-rajbhadurs.soeit@sanskriti.edu.in

A directory is a grouping of similar files on the hard drive. A directory may be thought of as a container that houses files and folders. We may keep all of a file's characteristics in a directory or just a few of them. Different types of files may be found in directories. We can retain the information pertaining to the files with the aid of the directory.

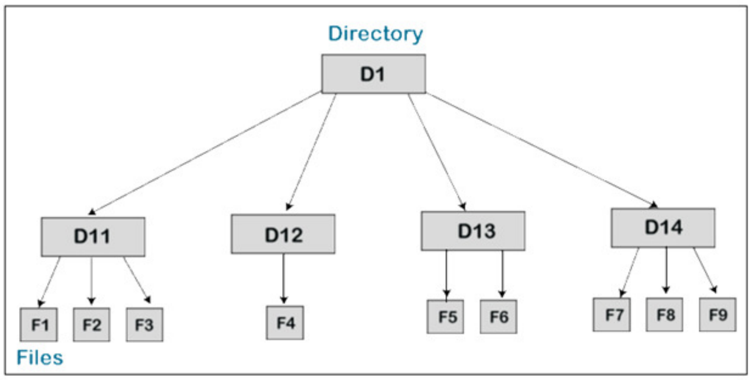


Figure: 12.1: Directory of hard drive

Partition the hard drive into many, varying-size partitions so that we can benefit from different file systems on different operating systems. Minidisks or volumes are other names for partitions.In each division, there has to be at least one directory. It allows us to list every file on the partition. There is a directory entry that is kept for each file in the directory, and it contains all the data pertinent to the file[1].

A. **Name:** The directory's name that the user may see is called "Name."
B. **Type:** A directory's type refers to its structure, such as whether it is a single-level directory, a two-level directory, a tree-structured directory, or an acyclic graph directory.
C. **Location:** The location of the device on which a file's header is stored is referred to as location.
D. **Size:** Size refers to the quantity of words, blocks, or bytes in the file.
E. **Position:** Position refers to where the next-read and next-write pointers are located.
F. **Protection:** Access control on the read, write, delete, and execute is referred to as protection.
G. **Usage:** Usage refers to the creation, modification, and access times, among other things.

H. **Mounting:** Mounting refers to the process of grafting a file system's root into another file system's existing tree.

Directory operations

The numerous kinds of directory operations include:

A. **Creation:** A directory is created with this procedure. The directory's name must to be distinctive.

B. **Eliminating:** If there is a file in the directory that we no longer require, we may remove it. If the directory is not needed, we may also delete it entirely. You may also remove a directory that is empty. A directory that solely contains dot and dot-dot characters is said to be empty.

C. **Searching:** Performing a search implies looking across a directory for a certain file or another directory.

D. **List a directory:** With this operation, we may get a list of all the files that are present in the directory. For each file in the list, we can also obtain the contents of the directory entry.

In order to read the list of all files in a directory, the directory must first be opened. Once we have read the directory, it must be closed in order to free up internal tablespace[2].

Directory Structure Types

Different kinds of directory structures exist:

One-Level Directory: - The simplest directory structure is a single-level directory. A single-level directory has only one directory, which is referred to as the root directory. All the files are contained in a single directory in a single-level directory, which makes it simple to grasp. The user cannot build subdirectories in this beneath the root directory.

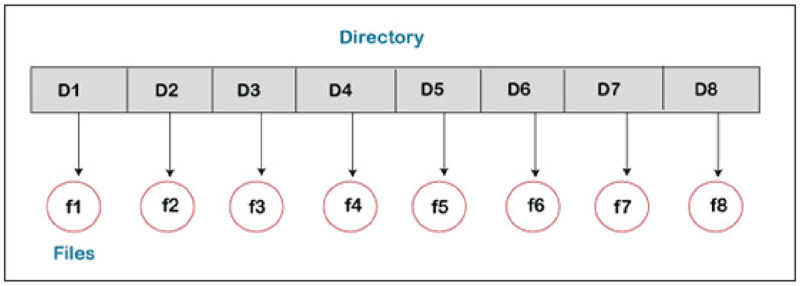


Figure: 12.2: Single-Level Directory

Single-Level Directory Benefits

The following are some benefits of a single-level directory:

A. Setting up a single-level directory is really simple.
B. If all the files in a single-level directory are tiny in size, it will be simple to search for them.
C. Operations like searching, creation, deletion, and updating may be done in a single-Level directory.

Single-Level Directory Drawbacks

The drawbacks of single-level directories include:

A. Searching will be challenging if the Single-Level Directory's directory size is huge.
B. Similar file types cannot be grouped in a single-level directory.

The probability of collision because two files cannot have the same name is a single-level directory's third drawback.Choosing an original file name is a challenging process.

Five-Level Directory

Another sort of directory layout is the two-level directory. Each user may have their own directory created with this feature. The two-level directory, which has a separate directory for each user, has one master node. Each of the users has their own directory present at the second level of the directory. No user may access another user's directory without their consent[3].

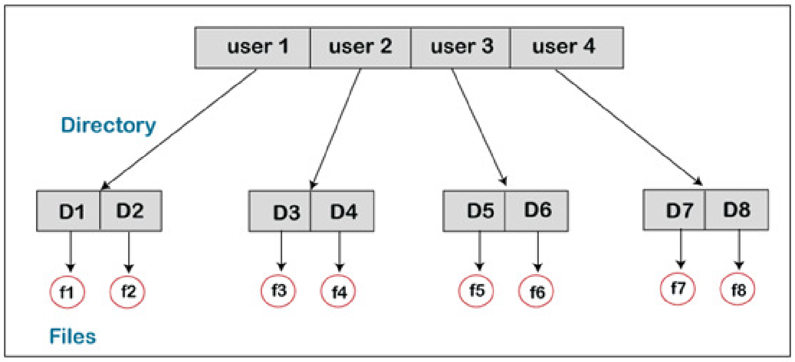


Figure: 12.3: Multi-Level Directory

Features of Two-Level Directories

The two-level directory has the following characteristics:

A. Different users' files may have the same name in a two-level directory.
B. Each file has a pathname, such /User-name/directory-name
C. In a two-level directory, we are unable to combine identically named files into a single directory for a particular user.
D. Searching is more efficient in a two-level directory since there is only one user's list that has to be browsed.

Two-Level Directory Benefits

The two-level directory has the following benefits: 1. Different users may share the same directory and file names.

File searches are made simple by the use of user groups and pathnames.

Cons of a Two-Level Directory

One user cannot share a file with another user in a two-level directory, which is one of its drawbacks.

The two-level directory also has the drawback of not being scalable.

Tree-Structured Directory

Another sort of directory structure is a tree-structured directory, in which a file or a subdirectory may be used as the directory entry. The two-level directory's restrictions are lessened by the tree-structured directory. The same kind of files may be compiled into a single directory.

Each user in a tree-structured directory has their own directory, and no user is permitted to access another user's directory. Although a user has access to root's data, they are unable to change or add to it. Only the root directory is accessible to the system administrator in full. We apply the present working notion in this and the searching is fairly efficient. There are two different types of paths we may use to access the file: absolute and relative[4].

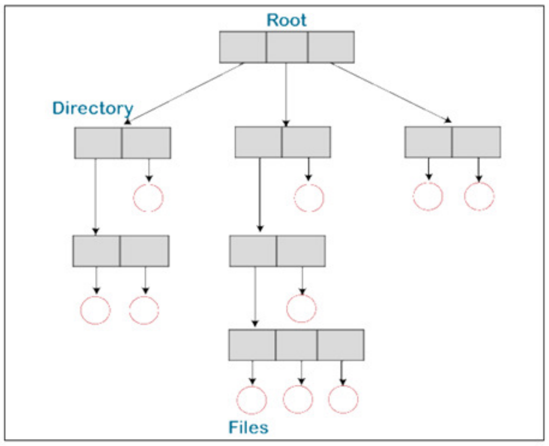


Figure: 12.4: Tree-Structured Directory

Tree-structured directories provide advantages.

The tree-structured directory has the following benefits:

A. It is very scalable.
B. There are less collisions in the tree-structures directory.
C. Because the absolute path and relative path may both be used in the tree-structure directory, searching is very simple.
D. Tree-Structure Directory Drawbacks

The following are the drawbacks of tree-structure directories:

A. Files cannot be shared in the tree-structure directory.
B. Tree-structure directories are inefficient because if we wish to access a file, it can be located in many different directories.
C. Each file does not fit into the hierarchal paradigm, which is another drawback of the tree-structure directory. The files need to be saved in separate folders.

Graph Acyclic Directory

The biggest issue with the tree-structure directory is file sharing since identical files cannot reside in different directories in the tree-structure directory. We can provide the sharing of files with the aid of the acyclic-graph directory. More than one directory in the acyclic-graph directory may lead to the same file or subfolder. These files may be shared across the two directory entries.

This kind of directory graph may be made with the use of aliases and links. An alternate path for the same file could potentially exist. Links come in two varieties: hard links (physical) and symbolic links (logical).

In the hard link (physical) example, we may erase the real files only if all references to the file are destroyed. If we delete the files in acyclic graph structures, then 1.In the symbolic link (logical) scenario, the file is simply deleted, leaving just a hanging point[5].

Acyclic-Graph Directory Benefits

The following are the benefits of the acyclic-graph directory:

File sharing inside the acyclic-graph directory is feasible.Searching is simple in the acyclic-graph directory due to different-different pathways.

Acyclic-Graph Directory Drawbacks

Acyclic-graph directories have the following drawbacks:

A. If files are linked together, deletion may cause an issue.
B. In the event that a soft link is used, all that is left once a file is erased is a hanging reference.
C. If we are utilising hard links in this situation, we must likewise erase any references to the file when we delete it.

Fourth General-Graph Directory

Another essential sort of directory structure is the General-Graph directory. This kind of directory allows us to establish cycles inside a directory from which we may deduce different directories by using more than one parent directory.Calculating the combined space or size that the directories and files take up is the fundamental problem with the general-graph directory.

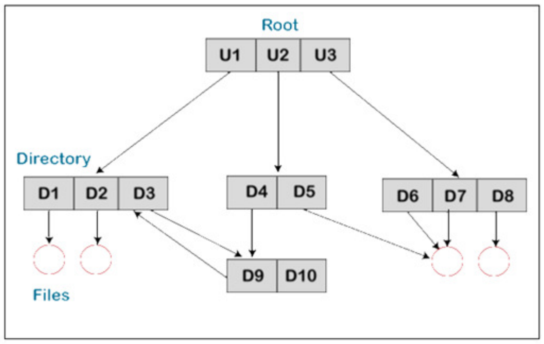


Figure: 12.5: General-Graph Directory

General-Graph directory benefits

- It is more versatile than other directory structures.
- The general-graph directory permits cycles.

General-Graph Directory drawbacks

- Garbage collection is necessary.
- Compared to alternative directory architectures, the general-graph directory is more expensive.

Implementation of a Directory

We use a variety of algorithms for implementing directories. Because it directly impacts system performance, choosing an appropriate method for directory implementation is a crucial effort. Based on the data structure, we may categories the method for implementing directories. We mainly use two categories of algorithms:

- A List, Linear
- Hash Table

List in linear form: The simplest simple approach used for directory implementation is the linear list. In this approach, all the files are kept in a directory in a manner similar to a singly linked list. Each file has a reference to the data blocks assigned to it as well as the next file in the directory[6].

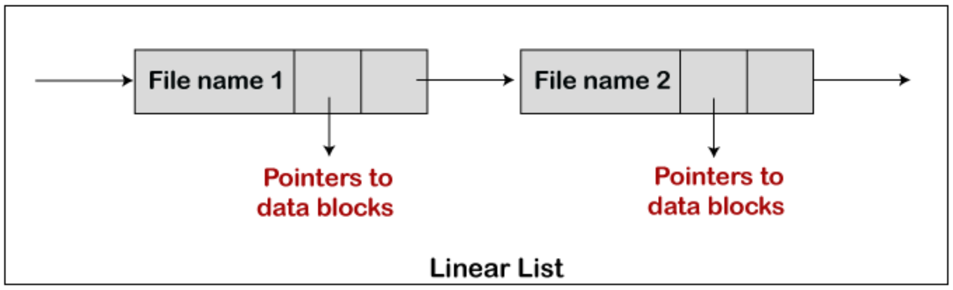


Figure: 12.6: Linear list implementation Directory

Features of a Linear List

The following are the characteristics of the linear list:

b The implementation of directories using single links has various drawbacks. We thus utilise a different technique called a hash table to overcome this issue. This approach combines a linked list with a hash table. Every file in a directory has a key-value pair that is formed for it, and we save that key-value pair in the hash table once it has been generated. We can identify the key and key points to a certain file that is stored in a directory by using the hash function on the file name. The work of searching in a linear list is challenging because we must search the full list in a linear list, but with a hash table technique, this necessity is not there. Therefore, searching in hash tables is fairly effective. We merely need to examine the hash table entries with the help of the key, and once we find an entry, we can use the value to get the matching file.

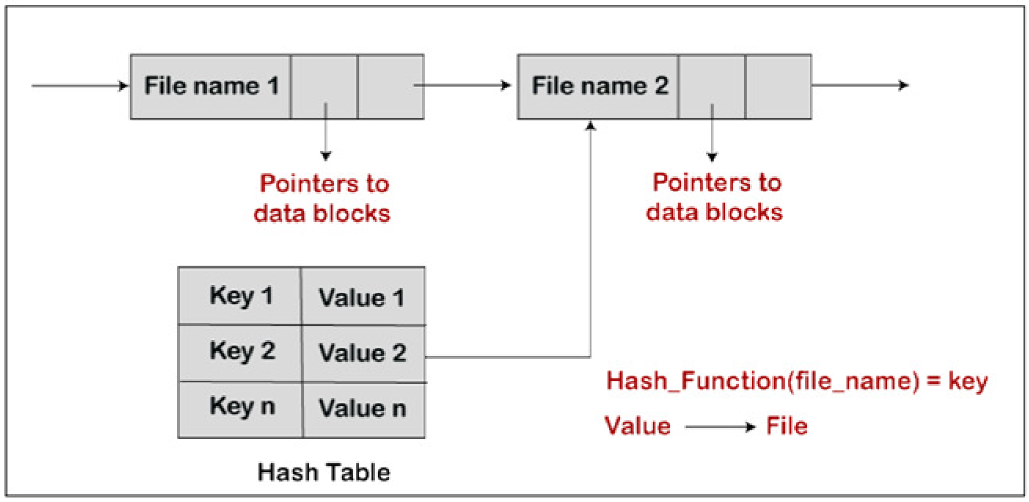


Figure: 12.7: Directory Structure

Directory Structure: Millions and thousands of users' worth of files may be found in a file system. These files are organised by the directory structure, which maintains entries for each associated file. File name, type, location, and the way that other system users may access the file are among the details included in the file entry.

The two key elements that need attention while managing a directory are listed below.

A. A user must be allowed to give a file any name they like, without having to worry about other system users using the same names.
B. A user must be able to view files made by other users as well as distribute files that he generates.

Both of the aforementioned qualities are provided by directory structures. A directory always contains details about the collection of connected files. The file system looks for the file's entry in the directory whenever a user or a process requests a file, and when a match is found, it gets the file's location from that[7].

Directory Structures Types

The most popular categories of directories, as well as their advantages and disadvantages, will be covered in this section.

A single level of the directory

The root directory is the sole directory in a single level directory system. Subdirectories beneath the root directory cannot be created by users. Only the root directory contains all of the files that were generated by different users. As seen in the picture below, the root directory contains all of the files F1, F2, F3, and F4 that were generated by the various users.

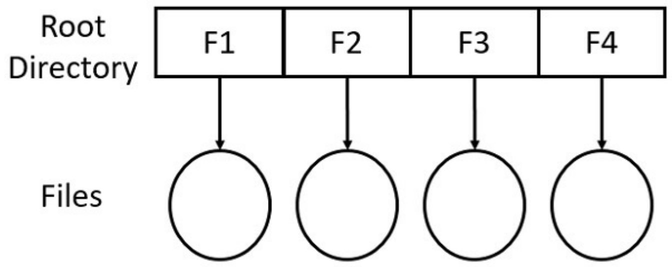


Figure: 12.8: Single-level Directory Structure

The single-level directory structure has one drawback: users cannot use the same file names as other users in the system. The old file will be deleted before the new one with the same name is created, even if the latter has the same name.

A directory with two levels

Users construct directories right within the root directory in a two-level directory structure. However, once a user creates such a directory, he is unable to construct any subdirectories within of it. As you can see in the diagram below, four users each established an own directory within the root directory. However, users don't create any subdirectories.

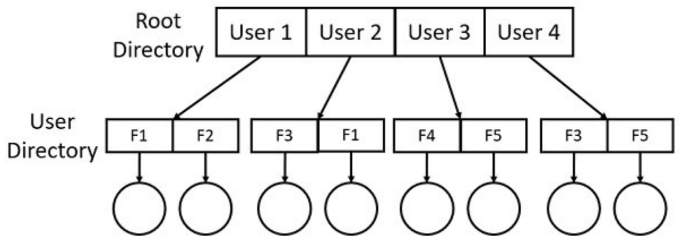


Figure: 12.9: Two-level Directory Structure

Each user may maintain their files independently inside of their own directory thanks to the two-level structure. The same name may be used for files located in separate user folders thanks to this arrangement.

Directory Structure in Hierarchy

Users may construct directories under the root directory and subdirectories within this structure in a hierarchical directory system. The user is allowed to construct as many sub-directories as desired, hence it is possible to build distinct sub-directories for various file kinds.

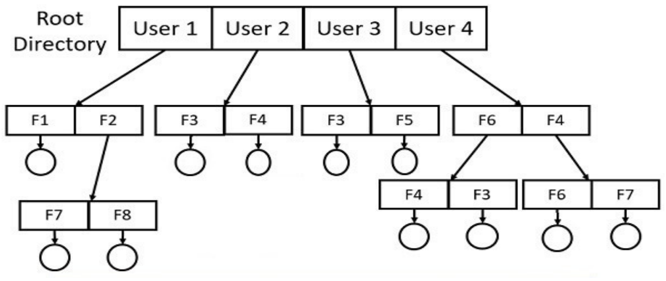


Figure: 12.10: Hierarchical Directory Structure

Here, the path is used to access the files according to their location. In this directory structure, there are two different ways to find the file.

Exactly Path

Here, the base directory—the root directory—is used to specify the path to the requested file.

Relational Path

Here, the base directory is either the user's directory or the directory where the required files are located.

Tree Directory Organization

Every directory and file in a tree directory structure, apart from the root directory, has only one parent directory. This full separation of the users allows for complete naming flexibility. Here, a user must go through two or more folders in order to view another user's file[8], [9].

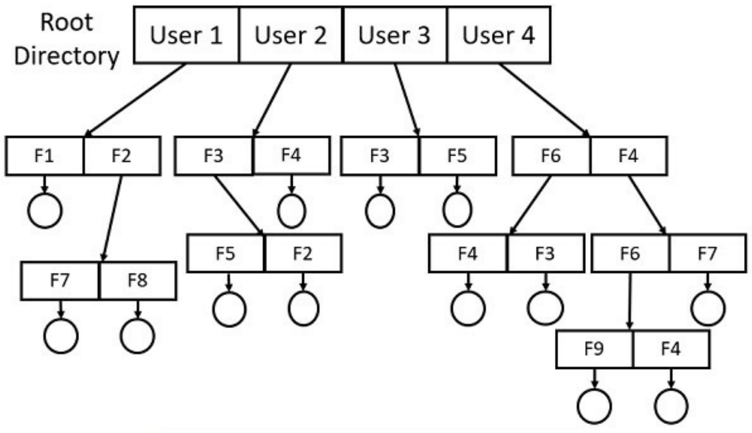


Figure: 12.11: Tree Directory Structure

Users may access shared files belonging to another user in an asymmetrical manner thanks to the tree directory structure. Users may access files in their own user directories, for instance, with shorter paths than other users.

The Directory Structure of Acyclic Graphs

The directory structure of an acyclic network may overcome this issue. A directory or file might have several parent directories thanks to this directory structure. As a result, other user directories that have access to a shared file in a directory may refer to it using links. See that the directory containing files F7 and F8 has two parent directories in the figure below.

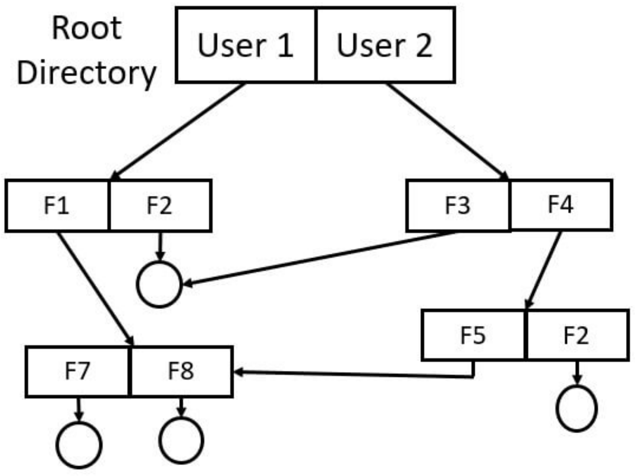


Figure: 12.12: Acyclic Directory Structure

Directory operations

A directory houses all of the connected files' entries, as was previously mentioned. The user must be able to add, remove, search, and list items in the directory in order to better organise it. The operation that can be carried out on the directory will be covered below.

Looking: may look in a directory for a specific file or another directory. To find all the files with the same name, it may also be searched.

Making: A new file may be made and added to the directory, or a new directory can be created, but it must have a name that is distinct inside that directory.

Erasing:A file may be removed from the directory if the user no longer need it. If it is not required, the whole directory may also be erased. You may also remove a directory that is empty. Dot and dotdot are used to represent an empty directory.

List an index: For each file in a list, it is possible to obtain a list of all the files in the directory as well as the contents of each entry. The directory must be opened in order to read the list of all the files within, and then it must be closed in order to release the internal tablespace.

Renaming: The name of a file or directory describes the information it contains and how it is used. If the contents or intended usage of a file changes, the file or directory may be renamed. The location of the file or directory inside the directory is also altered when it is renamed.

Link: The file may be permitted to appear in several directories. Here, the system call establishes a connection between the file and the name given by the path where the file should be located.

Disengage: The directory entry for a file that is unlinked and only found in one directory is deleted. Only the link is deleted if the file is present in many folders.

File System Mounting

The grouping of files in a file system structure that is accessible to the user of the group of users is referred to as mounting. It may be local or remote; in the former, disc drivers are connected as a single machine; in the latter, Network File System (NFS) is used to connect to directories on other computers so they can be utilised as if they were a part of the user's file system. Multiple volumes that are intended to be mounted in order to make them accessible inside the file-system namespace might be used to construct the directory structure. The process for mounting is straightforward; the file system attachment location and device name are provided to the OS together with the file structure location.

For instance, on a UNIX system, there is only one directory tree, and this tree must have a location for all of the available storage. To make the storage accessible, mounting is employed. The user's home directories may be stored on a file system mounted as /home, and they may be reached by using directory names that include the current date, such as /home/janc. Similar to how we would use /user/janc to access a file system mounted as /user. Then, by instructing the device driver to read the directory and check that it has the desired format, the operating system determines if the device includes a legitimate file system. When the file system is mounted at the designated mount point, the operating system ultimately records the directory structure. By using this technique, the operating system may navigate the directory tree and change file systems as necessary. A system may either permit multiple mounts of the same file system at various mount points or only one mount per file system. the Macintosh operating system, as an illustration. Every time the system comes into contact with a disc for the first time, it looks within the disc for the file system. If it finds one, it immediately mounts the system at the root level and adds a folder icon to the screen with the name of the file system written next to it. A two-level directory structure is maintained by the Microsoft OS[10].

Protection in File System

Users wish to safeguard the data kept in the file system against unauthorised access and physical harm. One may generate duplicate copies of the files to secure their information, and some systems will replicate the files automatically to prevent users from losing crucial data in the event that the original files are unintentionally deleted. The files may be harmed by hardware issues in circumstances including high temperatures, vandalism, head crashes, failures, etc.

Access types:

As numerous users have access to the same data, there is a need for security. File sharing is a means of granting users of the file system partial or full access. Access restrictions are one kind of file system protection, although this is an extreme case and is not useful in real life.Controlled access is what is necessary. Many other things might affect it. The operations that may be managed are as follows:

- Delete
- Append
- Execute
- Read
- Write
- List

However, only a minimal amount of protection is offered. For instance, it is possible to accomplish file copying by only sending a series of read requests. In this case, a user with read access may copy, print, and other actions on the file.

Access Management: There are several methods to access any file, but one of the popular ones is to provide all files and folders identity-dependent access. The identities of users and the types of access that have been given to them are included in a list called the access-control list. However, since every user must be mentioned, it is rather lengthy. This method may often be time-consuming and unfulfilling, particularly if the user list is not available prior to the assignment. The following categories are used to address this issue and reduce the size of the access-control list:

- The file's creator is its owner.
- Users who are sharing a file as a group.
- All other users make up the universe.

Implementation of File Systems in Operating Systems

A file is a grouping of connected data. The file system, which is placed on secondary storage, enables data to be saved, found, and retrieved, allowing for effective and practical access to the disc.

Layered organisational structure for files:

I/O Control level: Device drivers serve as an interface between hardware and the operating system and aid in data transmission between the hard drive and main memory. Block number is provided as the input, while low level hardware-specific instructions are provided as the output.

Basic file system: It sends device drivers generic instructions to read and write physical disc blocks. It controls the caches and memory buffers. The contents of a disc block may be stored in a block in a buffer, and cache holds frequently used file system information.

Module for file organisation -

It contains information about files, including their names, locations, and logical and physical building components. Physical blocks and logical blocks with numbers ranging from 0 to N do not match. The free space also keeps track of unallocated blocks.

Intelligent file system

It controls a file's metadata information, which entails all information about a file other than its actual contents. File control blocks are also used for maintenance. A file's owner, size, rights, and where the file's contents are located are all listed in the file control block (FCB).

Advantages:

- Code duplication is kept to a minimum.
- A logical file system may exist independently for each file system.

Cons:

- A high volume of concurrent file access leads to poor performance.

- Two different kinds of data structures may be used to create file systems:

First, on-disk structures

- They often include data on the overall amount of disc blocks, free disc blocks, where they are located, and other things. Different on-disk structures are listed below:
- The boot control block, which is often the first block on a drive, holds the data necessary to start an operating system.
- It is known as the boot block in UNIX and the partition boot sector in NTFS.
- Volume Control Block: It contains details about a certain partition, such as the number of free blocks, the size of blocks, and block pointers, etc.
- It is referred to as a super block in UNIX and a master file table in NTFS.
- File names and their corresponding inode numbers are stored in the directory structure.
- Includes file names and related file names in UNIX, and it is contained in the master file table in NTFS.

Per-File FCB: It includes information about each files and a special identification number that enables linkage with directory entry. It is kept in the master file table of NTFS.

Main memory-maintained in-memory structures: These are useful for file system administration and caching. The following list of in-memory structures:

- It gives details on each volume that has been mounted.
- The directory information of recently visited directories is stored in the directory-structure cache.
- The copy of FCB for each open file is included in the system-wide open-file table.

The per-process open-file table links to the relevant system wide open-file and holds the data that was opened by that specific process.

Implementing a directory:

Filenames containing pointers to the data blocks are maintained in a linear list. It takes a lot of time. To add a new file, we must first check the directory to make sure there are no other files with the same name before doing so. We look in the directory for the specified file and then delete it. Either by marking the directory entry as unused or by adding it to a list of open directories, we may reuse the entry. Using a value calculated from the file name, the hash table delivers a reference to the file. It speeds up directory searches. File insertion and deletion are simple processes. The main problem is that hash tables often have a predetermined size and are reliant on the size of the hash algorithm.

File System Structure

By making it easy to store, find, and retrieve data, file systems provide effective access to the disc. A file system must have the capacity to save, find, and retrieve files. The majority of operating systems use a layering strategy for all tasks, including file systems. Each layer of the file system is in charge of certain tasks. The picture below explains the several levels that make up the file system as well as the purpose of each layer.

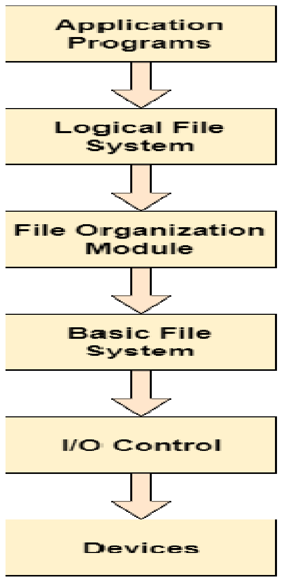


Figure: 12.13: File System Structure

File System Organization

The logical file system receives the first request from an application software when it requests for a file. The Meta data of the file and directory hierarchy are included in the logical file system. This layer will raise an error if the application software doesn't have the necessary file permissions. Logical file systems additionally check the file's path. Files are often broken up into several logical chunks. The hard drive is where files are kept and where they are retrieved. Various tracks and sectors make up a hard drive. Therefore, the logical blocks must be translated to physical blocks in order to store and retrieve the files. The File Organization Module does this mapping. It is also in charge of managing open space. Basic file system receives this information after File organization module determines which physical block the application software requires. The basic file system is in charge of sending the instructions to the I/O control to retrieve those blocks. I/O controllers hold the codes needed to access the hard drive. Device drivers are the name for these codes. Interrupt management is another responsibility of I/O controllers.

Directory implementation

The effectiveness, performance, and dependability of the file system are substantially impacted by the choice of directory-allocation and directory-management algorithms. We go through the trade-offs of selecting one of these algorithms in this section.

Linear List

A linear list of file names with pointers to the data blocks is the most basic way to construct a directory. Although it takes a while to perform, this approach is easy to programme. In order to create a new file, we must first check the directory to make sure that no other file with the identical name already exists. Then, we add a new entry to the directory's conclusion. We find the designated file in the directory and delete it by freeing up the space allotted to it. We may do one of three actions in order to reuse the directory entry. We may either link the item to a list of free directory entries or indicate it as unused (by giving it a special name, such as an all-blank name or by including a used-unused bit in each entry). A third option is to shorten the directory by copying the final item into the space that has been freed up. A linked list may be used to speed up the process of deleting a file. Finding a file involves a linear search, which is the true drawback of a linear list of directory entries. Users will notice if access to directory information is delayed since it is often utilised. In truth, a software cache is employed by many operating systems to keep the most current directory information. A cache hit eliminates the need to repeatedly read the data from disc. A sorted list facilitates binary searching and cuts down on search times. However, because we would need to transfer a lot of directory information to maintain a sorted directory, the need that the list be maintained sorted may make it more difficult to create and delete files. Here, a B-tree or another more advanced tree data structure would be useful. The ability to create a sorted directory listing without performing a separate sort operation is one benefit of the sorted list.

Table Hash

A hash table is yet another kind of data structure used for a file directory. The directory items are kept in a linear list using this approach, although a hash data structure is also used. A pointer to the file name in the linear list is returned by the hash table in exchange for a value it calculates from the file name. As a result, it may significantly cut down on directory search time. Although some consideration must be given for collisions—cases in which two file names hash to the same location—insertion and deletion are also reasonably simple.

A hash table's fixed size and dependency on that size for the hash function are its main drawbacks. Assume, for instance, that we create a linear-probing hash table with 64 entries. The residue of a division by 64, for example, is used by the hash function to transform file names into integers ranging from 0 to 63. In order to create a 65th file later, we would need to expand the directory hash table to, say, 128 entries. As a consequence, a new hash function that maps file names to values between 0 and 127 is required, and the directory entries need to be rearranged to reflect the new hash-function values. An alternative is to use a chained-overflow hash table. Instead of using distinct values for each hash element, we may use linked lists, which allows us to avoid collisions by simply adding the new entry to the linked list. Because looking for a name may entail going through a linked list of conflicting table entries, lookups may be a little slower. Even still, this approach is probably going to be a lot quicker than a linear search over the whole directory.

Allocation Procedures

The allocation of disc space for the files may be done in a variety of ways. The system's effectiveness and efficiency will be strongly impacted by the choice of a suitable allocation technique. The disc will be used and the files accessible according to the allocation mechanism. The following allocation techniques are available.

- Contiguous Allocation.
- Extents
- Linked Allocation
- Clustering
- FAT
- Indexed Allocation

- Linked Indexed Allocation
- Multilevel Indexed Allocation
- Inode

OS' Free Space Management

The term "file management system in an operating system" refers to the system software that controls and maintains track of free areas to allocate and de-allocate memory blocks to files. An operating system has a "free space list" that keeps track of the available blocks. When a file is generated, the operating system looks through the list of available free space for the needed space to store a file. The file system releases the specified space when a file is deleted and adds it to the "free space list."

Free Space Management Techniques in OS

The simultaneous allocation and de-allocation of memory blocks (managing free space) by an operating system is not a simple task. When creating new free space or clearing space after removing a file, the operating system employs a variety of techniques. The implementation of a free space list may be done in a number of ways. The following is an explanation of them: The most popular way to construct the free space list is using a bit vector. A "Bit map" is another name for a bit vector. Each bit in the collection or series corresponds to a disc block. The bits only accept values of 1 or 0. When the block bit is set to 1, it indicates that the block is empty, and when it is set to 0, it indicates that the block is not free. It is given to a few files. Since each bit in the bit vector represents 0, the initial state of all blocks is one of emptiness.

I/O interface for applications

Through structural methods and operating system interfaces in this part so that I/O devices may be handled consistently and uniformly. We describe, for example, how new discs and other devices may be introduced to a computer without causing the operating system to malfunction and how a program can access a file on a drive without knowing what sort of disc it is. The technique used in this case incorporates abstraction, encapsulation, and software layering, much as in other complicated software-engineering challenges. By defining a few broad types, we may specifically abstract away the specific variances in I/O devices. An interface, which is a standardised collection of functions, is used to access each generic type. Device drivers, which are kernel modules that are internally tuned to each device but expose one of the standard interfaces, include the differences. The software layering of the I/O-related parts of the kernel. Similar to how the I/O system calls encapsulate device functionality in a few generic classes that conceal hardware variations from applications, the device-driver layer's function is to hide the differences between device controllers from the I/O subsystem of the kernel. The task of the operating-system developer is made easier by separating the I/O subsystem from the hardware. The producers of hardware also profit from it. They either build device drivers to connect the new hardware to well-liked operating systems, or they design new devices to be compatible with an existing host controller interface (like SCSI-2) As a result, we are able to connect new peripherals to computers without having to wait for the operating-system vendor to provide support code. The device-driver interface has different standards for each kind of operating system, which is unfortunate for makers of device hardware. Multiple device drivers, such as those for MS-DOS, Windows 95/98, Windows NT/2000, and Solaris, may be included with a particular device upon launch.

Kernel I/O Subsystem

Interact via input and output (I/O) devices with the computer system. I/O refers to the movement of data between RAM and different I/O peripherals. Keyboards, mice, card readers, scanners, voice recognition systems, and touch displays are examples of input devices that we may use to enter data. By using output devices like speakers, plotters, printers, and monitors, we may get information from computers. These gadgets are not directly linked to the CPU. However, an interface is used to control the data transfers between them. System bus signals are converted via this interface into and out of the right format for the given device. These external devices and the CPU interact with one another using I/O registers. Many I/O services are offered by the kernel. The kernel offers a number of services including caching, scheduling, spooling, device reservation, and error handling that depend on the infrastructure of the hardware and device drivers.

Planning: The word "schedule" refers to choosing an ideal order to complete a string of I/O requests. Scheduling may improve the system's overall performance, equally distribute device access rights across all processes, and shorten the typical I/O completion times for waits, responses, and turnarounds. A blocked I/O system call is made by an application, and the request is then added to the wait queue for that device, which is managed by OS engineers.

Buffering: Data transferred between two devices or between a device and an application is briefly stored or kept in the buffer, a part of main memory, by these devices or applications. helps in addressing differences in device speed. Aids in resolving device transfer size inconsistencies. The kernel memory receives data that was previously in user application memory. To preserve "copy semantics," data is then transmitted to the device from kernel memory. It stops an application from changing a buffer's contents while it is being written.

Cache: It entails keeping a copy of the data somewhere that is simpler to access than the original. For instance, when you download a file from a Web page, it is placed in a cache subfolder beneath your browser's directory on your hard drive. The browser may receive files from the cache rather than the real server when you go back to a website you just viewed, saving you time and lessening network traffic. The difference between a cache and a buffer is that a cache saves a copy of a data item that already exists, while a buffer keeps two copies of the same data item.

SpoolingA device's spool is a buffer that stores tasks until the device is prepared to accept them. Disks are seen by spooling as a huge buffer that can retain as many jobs as the device need until the output devices are prepared to accept them. A buffer stores output for a device that cannot handle interleaved data streams if it can only process one request at a time. A user may inspect certain data streams via spooling and, if desired, remove them. Whenever you are utilising a printer, for instance.

Handling Errors

Operating systems with protected memory may protect against a variety of hardware and software flaws, preventing even the smallest mechanical error from leading to a system failure. Devices and I/O transfers may malfunction for a number of reasons, both temporary (such as when a network becomes overloaded) and persistent (such as when a disc controller fails). Calls to the I/O Protection System are necessary for I/O. User applications may attempt to obstruct normal operation by

Operating system streams

We must now ascertain what is on the other side of the buffer and file information block because even while the programme initiates all of the actions discussed in the previous chapter, they cannot be completed without the operating system's assistance. The operating system must offer the methods necessary to ensure that data transmission and file storage are handled correctly. These must be made available to all programmes, generally by requests for protection from supervisors. Therefore, the system must:

• Specify the format of the file information block and how it will relate to the way files are represented in programmes.

• Offer guidelines for carrying out the legal activities. For whatever information they want regarding the file, they often refer to the file information block, which is available during supervisor calls.

Networking programs with systems:

The file information block and buffer were used to handle the transactions between the programmes, and now it's time to think about what resources the operating system must make available to the programmes.

To complete the link, information about the actual stream or file that matches the specification provided by the programme must be obtained and combined with that information. Of course, if the programme requests something impractical (random access to a stream, a file that doesn't exist, a request to use a device that is already in use, etc.), the necessary error signals must be generated and communicated. This often entails creating some type of connection between the device descriptor, which provides details about the actual device to be utilised, and the program's file information block.

A connection may need to be made to another computer over a communications line, a tape may need to be installed and examined, and a plotter may need human attention to load fresh paper. Other tasks may also need to be performed on the device itself or on structures utilised in its administration. Since the nature and specifics of these actions vary depending on the device, it wouldn't make sense to permanently include them into the operating system (otherwise, you wouldn't be able to connect any new devices). Individual device drivers, which we will cover in more detail later, handle the needs of various devices instead. For the time being, it is sufficient that the system has a means of establishing the necessary connection. (Or, more accurately, that such a provision is necessary for the system to function rationally.)

Additional housekeeping chores, such as reading the initial buffer load and marking the device as in use, may be necessary depending on the configuration of the system.

Writing and reading:

These procedures essentially consist of keeping the buffer in the proper condition from the perspective of the system. An output buffer should be emptied when it is full and filled when the input buffer is empty. The system must make sure that information is transported to and from the devices as needed, even if the specifics of these operations rely on the device and are described by procedures in the device descriptors.

There may not be much more to the routine input and output in simple circumstances than is suggested by the preceding paragraph, although simple cases are not always the case. If so, the file information block often takes care of the issues; we go into more detail about these issues under the following item.

Operating System (OS) performance monitoring

Operating system, SQL Server, and database performance are all monitored as part of effective SQL Server performance management. We provided monitoring tools for the latter two performance measures in How to monitor your SQL Server instances and databases. We will discuss tools for operating system performance monitoring in this post. Monitoring the performance of the three groups stated above gives a comprehensive view of the system's health and essential data for identifying performance issues and bottlenecks. Metrics of operating system performance are linked to processor, memory, disc, and network performance. The available memory, average bytes per read/write, average read/write time, disc reads/writes per second, network utilisation, pages input per second, pages per second, processor queue length, and processor utilisation are some of the most crucial system performance measures. The monitoring objectives and performance specifications will determine the metrics that will be utilised. A DBA must be able to increase performance based on monitoring data, as well as spot possible problems and bottlenecks early on and resolve them before they have an impact on the system.

REFERENCES

[1] A. V. Gordeyev and A. V. Andreyev, “OpenLDAP directory service structure as a search filter,” Informatsionno-Upravliaiushchie Sist., 2019, doi: 10.31799/1684-8853-2019-2-52-56.

[2] S. A. Dragly et al., “Experimental directory structure (Exdir): An alternative to HDF5 without introducing a new file format,” Front. Neuroinform., 2018, doi: 10.3389/fninf.2018.00016.

[3] K. E. Taylor et al., “CMIP6 Global Attributes , DRS , Filenames , Directory Structure , and CV ’ s,” Syntax, 2017.

[4] H. Kawaji, “dirHub: a trackHub configurator with directory structure projection,” bioRxiv, 2018.

[5] R. Fernández-Pascual, A. Ros, and M. E. Acacio, “To be silent or not: on the impact of evictions of clean data in cache-coherent multicores,” J. Supercomput., 2017, doi: 10.1007/s11227-017-2026-6.

[6] J. Morim et al., “A global ensemble of ocean wave climate projections from CMIP5-driven models,” Sci. Data, 2020, doi: 10.1038/s41597-020-0446-2.

[7] R. Titos-Gil et al., “Way Combination for an Adaptive and Scalable Coherence Directory,” IEEE Trans. Parallel Distrib. Syst., 2019, doi: 10.1109/TPDS.2019.2917185.

[8] A. Robbins, “UNIX Tutorial for Beginners,” October, 2010.

[9] A. Ros, M. E. Acacio, and J. M. García, “A scalable organization for distributed directories,” J. Syst. Archit., 2010, doi: 10.1016/j.sysarc.2009.11.006.

[10] C. White et al., “dsmcFoam+: An OpenFOAM based direct simulation Monte Carlo solver,” Comput. Phys. Commun., 2018, doi: 10.1016/j.cpc.2017.09.030.

Questionnaire

1. Describe the various elements of a computer system and explain the distinctive features of a computer system and manual system.
2. What are the evolution of operating system?
3. What's the main purpose of an operating system?
4. What is the role of process management in operating system?
5. What are the different operating systems?
6. What are the different kinds of operations that are possible on semaphore?
7. What is the Process of Contiguous Memory Allocation?
8. What is the role of memory in operating system?
9. How do you detect deadlock in operating system?
10. What is file system interface in operating system?
11. What is the purpose of swap space management?
12. What is Swap-Space Management?
13. What is Kernel I/O Subsystem?
