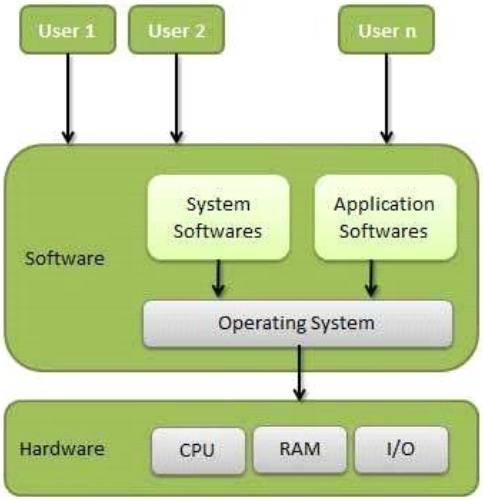
06 July 2021 00:54

1. What is the main purpose of an operating system?

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.
Some popular Operating Systems include Linux Operating System, Windows Operating System, VMS, OS/400, AIX, z/OS, etc.

Definition

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



Following are some of important functions of an operating System.

- Memory Management
- Processor Management
- Device Management
- File Management
- Security
- Control over system performance
- Job accounting
- Error detecting aids
- Coordination between other software and users

Memory Management

Memory management refers to management of Primary Memory or Main Memory. Main memory is a large array of words or bytes where each word or byte has its own address.
Main memory provides a fast storage that can be accessed directly by the CPU. For a program to be executed, it must in the main memory. An Operating System does the following activities for memory management –

- Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.
- In multiprogramming, the OS decides which process will get memory when and how much.
- Allocates the memory when a process requests it to do so.
- De-allocates the memory when a process no longer needs it or has been terminated.

Processor Management

In multiprogramming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management –

- Keeps tracks of processor and status of process. The program responsible for this task is known as traffic controller.
- Allocates the processor (CPU) to a process.
- De-allocates processor when a process is no longer required.

Device Management

An Operating System manages device communication via their respective drivers. It does the following activities for device management –

- Keeps tracks of all devices. Program responsible for this task is known as the I/O controller.
- Decides which process gets the device when and for how much time.
- Allocates the device in the efficient way.
- De-allocates devices.

File Management

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.
An Operating System does the following activities for file management –

- Keeps track of information, location, uses, status etc. The collective facilities are often known as file system.
- Decides who gets the resources.
- Allocates the resources.
- De-allocates the resources.

Other Important Activities

Following are some of the important activities that an Operating System performs –

- Security – By means of password and similar other techniques, it prevents unauthorized access to programs and data.
- Control over system performance – Recording delays between request for a service and response from the system.
- Job accounting – Keeping track of time and resources used by various jobs and users.
- Error detecting aids – Production of dumps, traces, error messages, and other debugging and error detecting aids.
- Coordination between other softwares and users – Coordination and assignment of compilers, interpreters, assemblers and other software to the various users of the computer systems.

From <https://www.tutorialspoint.com/operating_system/os_overview.htm>

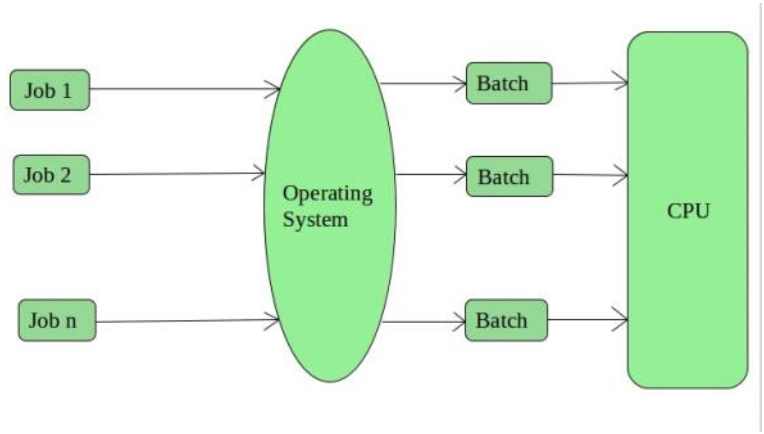
DISCUSS the type of os

An Operating System performs all the basic tasks like managing files, processes, and memory. Thus operating system acts as the manager of all the resources, i.e. **resource manager**. Thus, the operating system becomes an interface between user and machine.

Types of Operating Systems: Some widely used operating systems are as follows-

1. Batch Operating System –

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirement and group them into batches. It is the responsibility of the operator to sort jobs with similar needs.



Advantages of Batch Operating System:

- It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

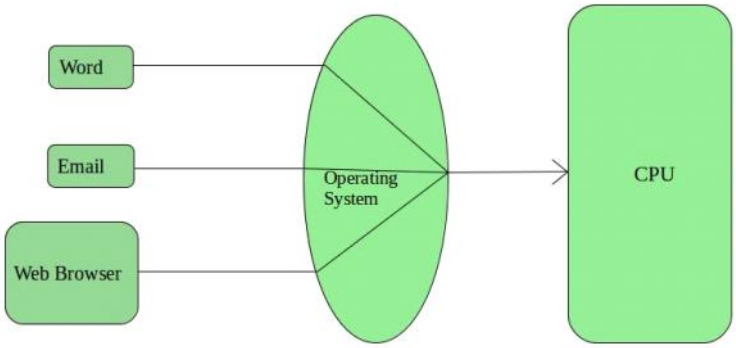
Disadvantages of Batch Operating System:

- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System: Payroll System, Bank Statements, etc.

2. Time-Sharing Operating Systems –

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also. The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.



Advantages of Time-Sharing OS:

- Each task gets an equal opportunity
- Fewer chances of duplication of software
- CPU idle time can be reduced

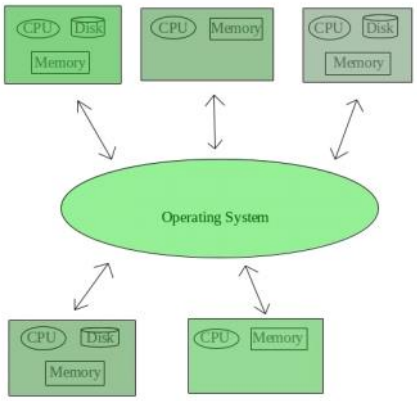
Disadvantages of Time-Sharing OS:

- Reliability problem
- One must have to take care of the security and integrity of user programs and data
- Data communication problem

Examples of Time-Sharing OSs are: Multics, Unix, etc.

3. Distributed Operating System –

These types of the operating system is a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, with a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as **loosely coupled systems** or distributed systems. These system's processors differ in size and function. The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.



Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

Disadvantages of Distributed Operating System:

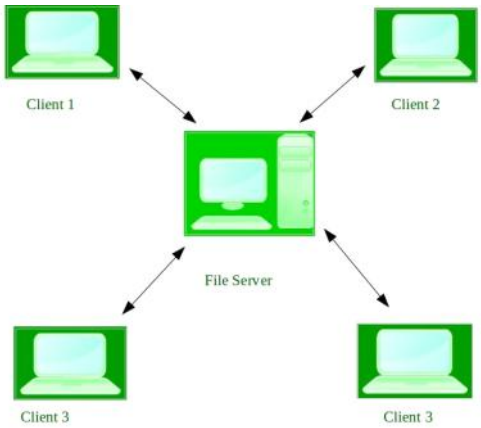
- Failure of the main network will stop the entire communication
- To establish distributed systems the language which is used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

Examples of Distributed Operating System are- LOCUS, etc.

4. Network Operating System –

These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions. These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network. One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the

network, their individual connections, etc. and that's why these computers are popularly known as **tightly coupled systems**.



Advantages of Network Operating System:

- Highly stable centralized servers
- Security concerns are handled through servers
- New technologies and hardware up-gradation are easily integrated into the system
- Server access is possible remotely from different locations and types of systems

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on a central location for most operations
- Maintenance and updates are required regularly

Examples of Network Operating System are: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

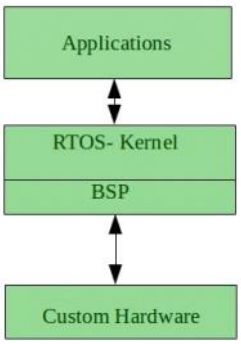
5. Real-Time Operating System –

These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

Real-time systems are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

Two types of Real-Time Operating System which are as follows:

- **Hard Real-Time Systems:**
  These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable. These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of any accident. Virtual memory is rarely found in these systems.
- **Soft Real-Time Systems:**
  These OSs are for applications where for time-constraint is less strict.



Advantages of RTOS:

- **Maximum Consumption:** Maximum utilization of devices and system, thus more output from all the resources
- **Task Shifting:** The time assigned for shifting tasks in these systems are very less. For example, in older systems, it takes about 10 microseconds in shifting one task to another, and in the latest systems, it takes 3 microseconds.
- **Focus on Application:** Focus on running applications and less importance to applications which are in the queue.
- Real-time **operating system in** the **embedded system:** Since the size of programs are small, RTOS can also be used in embedded systems like in transport and others.
- **Error Free:** These types of systems are error-free.
- **Memory Allocation:** Memory allocation is best managed in these types of systems.

Disadvantages of RTOS:

- **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on few applications to avoid errors.
- **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms:** The algorithms are very complex and difficult for the designer to write on.
- **Device driver and interrupt signals:** It needs specific device drivers and interrupts signals to respond earliest to interrupts.
- **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.
Examples of Real-Time Operating Systems are: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

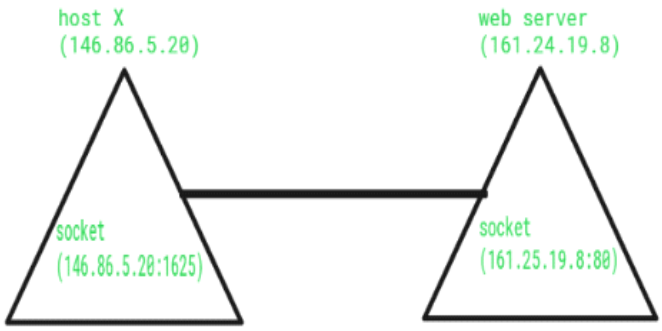
What is socket in OS ??
Socket in Computer Network

- Difficulty Level : Easy
- Last Updated : 10 May, 2020

A **socket** is one endpoint of a **two way** communication link between two programs running on the network. The socket mechanism provides a means of inter-process communication (IPC) by establishing named contact points between which the communication take place.

Like 'Pipe' is used to create pipes and sockets is created using **'socket'** system call. The socket provides bidirectional **FIFO** Communication facility over the network. A socket connecting to the network is created at each end of the communication. Each socket has a specific address. This address is composed of an IP address and a port number.

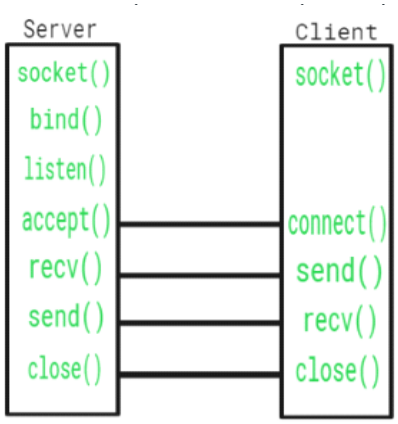
Socket are generally employed in client server applications. The server creates a socket, attaches it to a network port addresses then waits for the client to contact it. The client creates a socket and then attempts to connect to the server socket. When the connection is established, transfer of data takes place.



Types of Sockets :

There are two types of Sockets: the **datagram** socket and the **stream** socket.

1. **Datagram Socket :**
This is a type of network which has connection less point for sending and receiving packets. It is similar to mailbox. The letters (data) posted into the box are collected and delivered (transmitted) to a letterbox (receiving socket).
2. **Stream Socket**
In Computer operating system, a stream socket is type of interprocess communications socket or network socket which provides a connection-oriented, sequenced, and unique flow of data without record boundaries with well defined mechanisms for creating and destroying connections and for detecting errors. It is similar to phone. A connection is established between the phones (two ends) and a conversation (transfer of data) takes place.

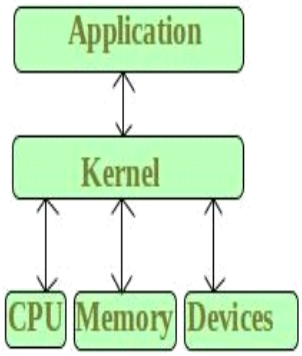


Function Call	Description
Create()	To create a socket
Bind()	It's a socket identification like a telephone number to contact
Listen()	Ready to receive a connection
Connect()	Ready to act as a sender
Accept()	Confirmation, it is like accepting to receive a call from a sender
Write()	To send data
Read()	To receive data
Close()	To close a connection

From <https://www.geeksforgeeks.org/socket-in-computer-network/>

WHAT IS MICROKERNAL IN OS ??
Microkernel in Operating Systems

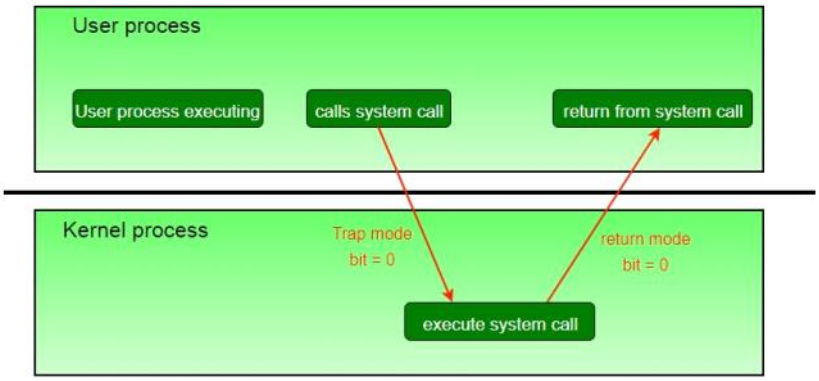
Kernel is the core part of an operating system which manages system resources. It also acts like a bridge between application and hardware of the computer. It is one of the first programs loaded on start-up (after the Bootloader).



Kernel mode and User mode of CPU operation

The CPU can execute certain instruction only when it is in the kernel mode. These instruction are called privilege instruction. They allow implementation of special operation whose execution by the user program could interface with the functioning of operating system or activity of another user program. For example, instruction for managing memory protection.

- The operating system puts the CPU in kernel mode when it is executing in the kernel so, that kernel can execute some special operation.
- The operating system puts the CPU in user mode when a user program is in execution so, that user program cannot interface with the operating system program.
- User-level instruction does not require special privilege. Example are ADD,PUSH,etc.



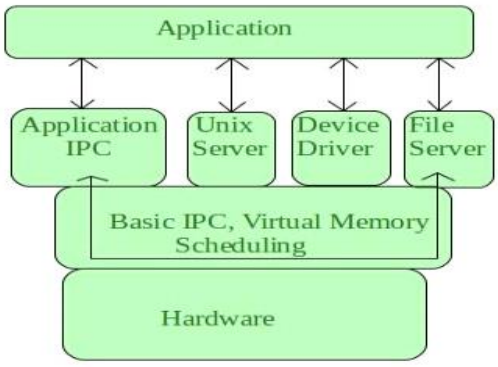
The concept of modes can be extended beyond two, requiring more than a single mode bit CPUs that support virtualization use one of these extra bits to indicate when the virtual machine manager, VMM, is in control of the system. The VMM has more privileges than ordinary user programs, but not so many as the full kernel.

System calls are typically implemented in the form of software interrupts, which causes the hardware's interrupt handler to transfer control over to an appropriate interrupt handler, which is part of the operating system, switching the mode bit to kernel mode in the process. The interrupt handler checks exactly which interrupt was generated, checks additional parameters (generally passed through registers) if appropriate, and then calls the appropriate kernel service routine to handle the service requested by the system call.

User programs' attempts to execute illegal instructions (privileged or non-existent instructions), or to access forbidden memory areas, also generate software interrupts, which are trapped by the interrupt handler and control is transferred to the OS, which issues an appropriate error message, possibly dumps data to a log (core) file for later analysis, and then terminates the offending program.

What is Microkernel?

Microkernel is one of the classification of the kernel. Being a kernel it manages all system resources. But in a microkernel, the **user services** and **kernel services** are implemented in different address space. The user services are kept in **user address space**, and kernel services are kept under **kernel address space**, thus also reduces the size of kernel and size of operating system as well.



It provides minimal services of process and memory management. The communication between client program/application and services running in user address space is established through message passing, reducing the speed of execution microkernel. The Operating System **remains unaffected** as user services and kernel services are isolated so if any user service fails it does not affect kernel service. Thus it adds to one of the advantages in a microkernel. It is easily **extendable** i.e. if any new services are to be added they are added to user address space and hence requires no modification in kernel space. It is also portable, secure and reliable.

Microkernel Architecture –

Since kernel is the core part of the operating system, so it is meant for handling the most important services only. Thus in this architecture only the most important services are inside kernel and rest of the OS services are present inside system application program. Thus users are able to interact with those not-so important services within the system application. And the microkernel is solely responsible for the most important services of operating system they are named as follows:

- Inter process-Communication

- Memory Management
- CPU-Scheduling

Advantages of Microkernel –

- The architecture of this kernel is small and isolated hence it can function better.
- Expansion of the system is easier, it is simply added in the system application without disturbing the kernel.

Eclipse IDE is a good example of Microkernel Architecture

From <https://www.geeksforgeeks.org/microkernel-in-operating-systems/>

Difference between Process and Thread ??

Process:

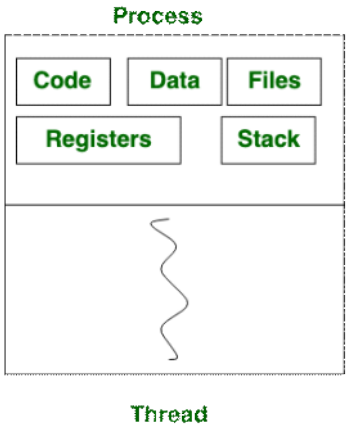
Process means any program is in execution. Process control block controls the operation of any process. Process control block contains information about processes for example Process priority, process id, process state, CPU, register, etc. A process can creates other processes which are known as **Child Processes**. Process takes more time to terminate and it is isolated means it does not share memory with any other process.

The process can have the following states like new, ready, running, waiting, terminated, suspended.

Thread:

Thread is the segment of a process means a process can have multiple threads and these multiple threads are contained within a process. A thread have 3 states: running, ready, and blocked.

Thread takes less time to terminate as compared to process and like process threads do not isolate.



Difference between Process and Thread:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Process consume more resources.	Thread consume less resources.
7.	Process is isolated.	Threads share memory.
8.	Process is called heavy weight	Thread is called light weight process.

	process.	
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.
10.	If one process is blocked then it will not effect the execution of other process	Second thread in the same task couldnot run, while one server thread is blocked.
11.	Process has its own Process Control Block, Stack and Address Space.	Thread has Parents' PCB, its own Thread Control Block and Stack and common Address space.

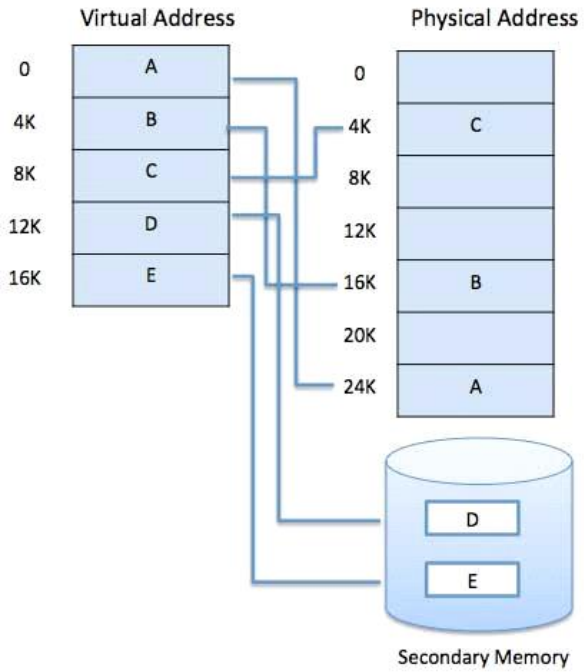
From <https://www.geeksforgeeks.org/difference-between-process-and-thread/>

WHAT IS VIRTUAL MEMORY ???

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory and it is a section of a hard disk that's set up to emulate the computer's RAM. The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address. Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

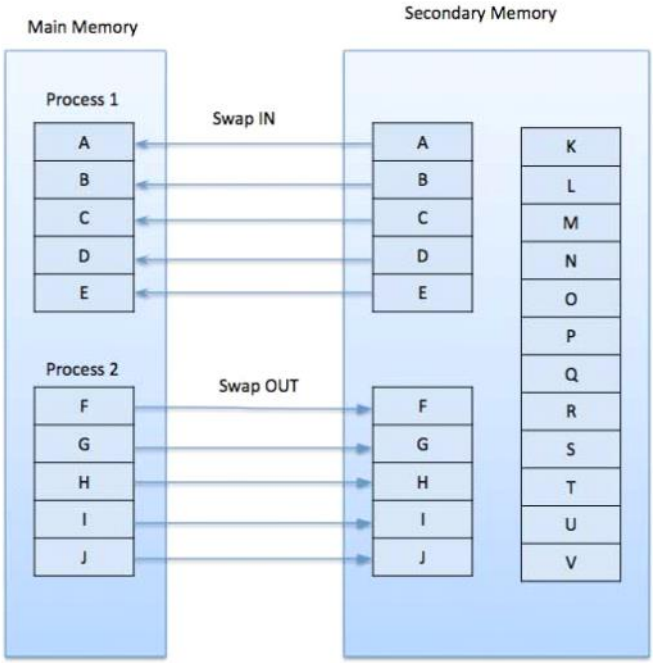
Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware. The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below –



Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

Demand Paging

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance. When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.



While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a page fault and transfers control from the program to the operating system to demand the page back into the memory.

Advantages

Following are the advantages of Demand Paging –

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

Page Replacement Algorithm

Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String

The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

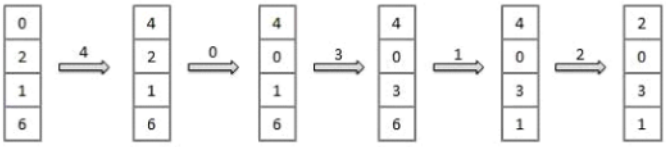
- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page p, then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses – 123,215,600,1234,76,96
- If page size is 100, then the reference string is 1,2,6,12,0,0

First In First Out (FIFO) algorithm

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



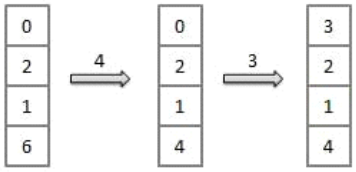
Fault Rate = 9 / 12 = 0.75

Optimal Page algorithm

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x



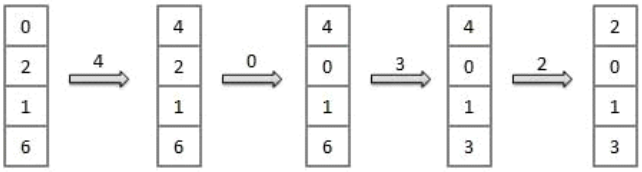
Fault Rate = 6 / 12 = 0.50

Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x



Fault Rate = 8 / 12 = 0.67

Page Buffering algorithm

- To get a process start quickly, keep a pool of free frames.
- On page fault, select a page to be replaced.
- Write the new page in the frame of free pool, mark the page table and restart the process.
- Now write the dirty page out of disk and place the frame holding replaced page in free pool.

Least frequently Used(LFU) algorithm

- The page with the smallest count is the one which will be selected for replacement.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.

Most frequently Used(MFU) algorithm

- This algorithm is based on the argument that the page with the smallest count was probably just brought in and has yet to be used.

From <https://www.tutorialspoint.com/operating_system/os_virtual_memory.htm>

WHAT IS thrashing ???

To know what is thrashing, you must first be aware of swapping and page fault. So lets start with those concepts:

First we need to know what actually happens inside your OS that makes your system behave as if it has a very large memory. In operating systems that implement a virtual memory space, the programs allocate memory from an address space that may be much larger than the actual amount of RAM the system possesses. OS decides which program's memory is actually in RAM at any specific instant of time.

Page Fault and Swapping: A page fault occurs when the memory access requested (from the virtual address space) does not map to something that is in RAM. A page must then be sent from RAM to swap, so that the requested new page can be brought from swap to RAM. This results in 2 disk I/Os. Now you might know that disk I/Os are very slow as compared to memory access.

Thrashing: Now if it happens that your system has to swap pages at such a higher rate that **major chunk of CPU time is spent in swapping** then this state is known as thrashing. So effectively during thrashing, the CPU spends less time in some actual productive work and more time in swapping.

Now the effects of thrashing and also the extent to which thrashing occurs will be decided by the type of page replacement policy.

1. **Global Page Replacement**: The paging algorithm is applied to all the pages of the memory regardless of which process "owns" them. A page fault in one process may cause a replacement from any process in memory. Thus, the size of a partition may vary randomly.

2. **Local Page Replacement**: The memory is divided into partitions of a predetermined size for each process and the paging algorithm is applied independently for each region. A process can only use pages in its partition.

What happens after Thrashing starts?

If **global** page replacement is used, situations worsens very quickly. CPU thinks that CPU utilization is decreasing, so it tries to increase the degree of multiprogramming. Hence bringing more processes inside memory, which in effect increases the thrashing and brings down CPU utilization further down. The CPU notices that utilization is going further down, so it increases the degree of multiprogramming further and the cycle continues.

The solution can be **local** page replacement where a process can only be allocated pages in its own region in memory. If the swaps of a process increase also, the overall CPU utilization does not decrease much. If other transactions have enough page frames in the partitions they occupy, they will continue to be processed efficiently.

But local page replacement has its own disadvantages which you can now explore further.

From <https://practice.geeksforgeeks.org/problems/thrashing-in-os>

What is a thrad ??

Thread in Operating System

- Difficulty Level : Medium
- Last Updated : 28 Jun, 2021

What is a Thread?

A thread is a path of execution within a process. A process can contain multiple threads.

Why Multithreading?

A thread is also known as lightweight process. The idea is to achieve parallelism by dividing a process into multiple threads. For example, in a browser, multiple tabs can be different threads. MS Word uses multiple threads: one thread to format the text, another thread to process inputs, etc. More advantages of multithreading are discussed below

Process vs Thread?

The primary difference is that threads within the same process run in a shared memory space, while processes run in separate memory spaces.

Threads are not independent of one another like processes are, and as a result threads share with other threads their code section, data section, and OS resources (like open files and signals). But, like process, a thread has its own program counter (PC), register set, and stack space.

Advantages of Thread over Process

1. Responsiveness: If the process is divided into multiple threads, if one thread completes its execution, then its output can be immediately returned.

2. Faster context switch: Context switch time between threads is lower compared to process context switch. Process context switching requires more overhead from the CPU.

3. Effective utilization of multiprocessor system: If we have multiple threads in a single process, then we can schedule multiple threads on multiple processor. This will make process execution faster.

4. Resource sharing: Resources like code, data, and files can be shared among all threads within a process.

Note: stack and registers can't be shared among the threads. Each thread has its own stack and registers.

5. Communication: Communication between multiple threads is easier, as the threads shares common address space. while in process we have to follow some specific communication technique for communication between two process.

6. Enhanced throughput of the system: If a process is divided into multiple threads, and each thread function is considered as one job, then the number of jobs completed per unit of time is increased, thus increasing the throughput of the system.

Types of Threads

There are two types of threads.

User Level Thread

Kernel Level Thread

From <https://www.geeksforgeeks.org/thread-in-operating-system/>

RAID (Redundant Arrays of Independent Disks)

- Difficulty Level : Easy
- Last Updated : 20 Aug, 2019

RAID, or "Redundant Arrays of Independent Disks" is a technique which makes

use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both. The term was coined by David Patterson, Garth A. Gibson, and Randy Katz at the University of California, Berkeley in 1987.

Why data redundancy?

Data redundancy, although taking up extra space, adds to disk reliability. This means, in case of disk failure, if the same data is also backed up onto another disk, we can retrieve the data and go on with the operation. On the other hand, if the data is spread across just multiple disks without the RAID technique, the loss of a single disk can affect the entire data.

Key evaluation points for a RAID System

- **Reliability:** How many disk faults can the system tolerate?
- **Availability:** What fraction of the total session time is a system in uptime mode, i.e. how available is the system for actual use?
- **Performance:** How good is the response time? How high is the throughput (rate of processing work)? Note that performance contains a lot of parameters and not just the two.
- **Capacity:** Given a set of N disks each with B blocks, how much useful capacity is available to the user?
  RAID is very transparent to the underlying system. This means, to the host system, it appears as a single big disk presenting itself as a linear array of blocks. This allows older technologies to be replaced by RAID without making too many changes in the existing code.

Different RAID levels

RAID-0 (Striping)

- Blocks are “striped” across disks.

Disk 0	Disk 1	Disk 2	Disk 3
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

In the figure, blocks “0,1,2,3” form a stripe.

- Instead of placing just one block into a disk at a time, we can work with two (or more) blocks placed into a disk before moving on to the next one.

Disk 0	Disk 1	Disk 2	Disk 3
0	3	4	6
1	3	5	7
8	10	12	14
9	11	13	15

Evaluation:

- Reliability: 0
  There is no duplication of data. Hence, a block once lost cannot be recovered.
- Capacity: N*B
  The entire space is being used to store data. Since there is no duplication, N disks each having B blocks are fully utilized.

RAID-1 (Mirroring)

- More than one copy of each block is stored in a separate disk. Thus, every block has two (or more) copies, lying on different disks.

Disk 0	Disk 1	Disk 2	Disk 3
0	0	1	1
2	2	3	3
4	4	5	5
6	6	7	7

The above figure shows a RAID-1 system with mirroring level 2.

- RAID 0 was unable to tolerate any disk failure. But RAID 1 is capable of reliability.
  Evaluation:

  Assume a RAID system with mirroring level 2.

- Reliability: 1 to N/2
  1 disk failure can be handled for certain, because blocks of that disk would have duplicates on some other disk. If we are lucky enough and disks 0 and 2 fail, then again this can be handled as the blocks of these disks have duplicates on disks 1 and 3. So, in the best case, N/2 disk failures can be handled.
- Capacity: N*B/2
  Only half the space is being used to store data. The other half is just a mirror to the already stored data.
  RAID-4 (Block-Level Striping with Dedicated Parity)
- Instead of duplicating data, this adopts a parity-based approach.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
4	5	6	7	P1
8	9	10	11	P2
12	13	14	15	P3

In the figure, we can observe one column (disk) dedicated to parity.

- Parity is calculated using a simple XOR function. If the data bits are 0,0,0,1 the parity bit is XOR(0,0,0,1) = 1. If the data bits are 0,1,1,0 the parity bit is XOR(0,1,1,0) = 0. A simple approach is that even number of ones results in parity 0, and an odd number of ones results in parity 1.

C1	C2	C3	C4	Parity
0	0	0	1	1
0	1	1	0	0

Assume that in the above figure, C3 is lost due to some disk failure. Then, we can recompute the data bit stored in C3 by looking at the values of all the other columns and the parity bit. This allows us to recover lost data.

Evaluation:

- Reliability: 1
  RAID-4 allows recovery of at most 1 disk failure (because of the way parity works). If more than one disk fails, there is no way to recover the data.
- Capacity: (N-1)*B
  One disk in the system is reserved for storing the parity. Hence, (N-1) disks are made available for data storage, each disk having B blocks.
  RAID-5 (Block-Level Striping with Distributed Parity)
- This is a slight modification of the RAID-4 system where the only difference is that the parity rotates among the drives.

Disk 0	Disk 1	Disk 2	Disk 3	Disk 4
0	1	2	3	P0
5	6	7	P1	4
10	11	P2	8	9
15	P3	12	13	14
P4	16	17	18	19

In the figure, we can notice how the parity bit “rotates”.

- This was introduced to make the random write performance better.
  Evaluation:
- Reliability: 1
  RAID-5 allows recovery of at most 1 disk failure (because of the way parity works). If more than one disk fails, there is no way to recover the data. This is identical to RAID-4.
- Capacity: (N-1)*B
  Overall, space equivalent to one disk is utilized in storing the parity. Hence, (N-1) disks are made available for data storage, each disk having B blocks.
  What about the other RAID levels?

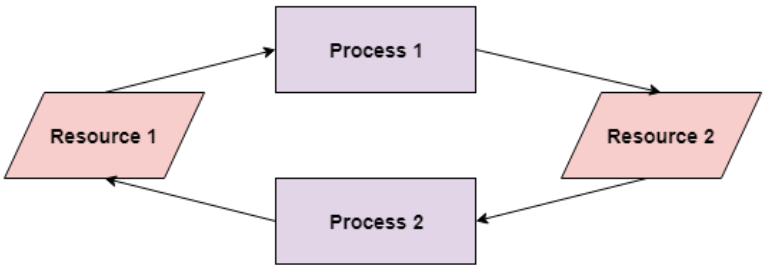
  RAID-2 consists of bit-level striping using a Hamming Code parity. RAID-3 consists of byte-level striping with a dedicated parity. These two are less commonly used.

  RAID-6 is a recent advancement which contains a distributed double parity, which involves block-level striping with 2 parity bits instead of just 1 distributed across all the disks. There are also hybrid RAIDs, which make use of more than one RAID levels nested one after the other, to fulfill specific requirements.

From <https://www.geeksforgeeks.org/raid-redundant-arrays-of-independent-disks/>

What is a deadlock ? Different conditions to achieve a deadlock.

A deadlock happens in operating system when two or more processes need some resource to complete their execution that is held by the other process.



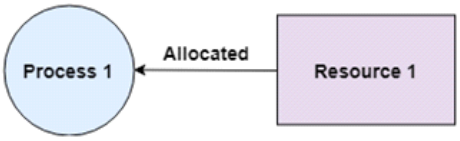
Deadlock in Operating System

In the above diagram, the process 1 has resource 1 and needs to acquire resource 2. Similarly process 2 has resource 2 and needs to acquire resource 1. Process 1 and process 2 are in deadlock as each of them needs the other's resource to complete their execution but neither of them is willing to relinquish their resources.

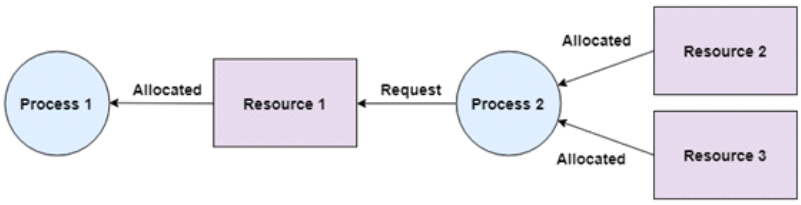
Coffman Conditions

A deadlock occurs if the four Coffman conditions hold true. But these conditions are not mutually exclusive.
The Coffman conditions are given as follows –

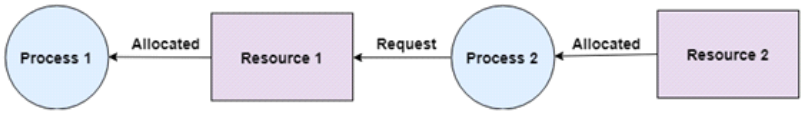
- **Mutual Exclusion**
  There should be a resource that can only be held by one process at a time. In the diagram below, there is a single instance of Resource 1 and it is held by Process 1 only.



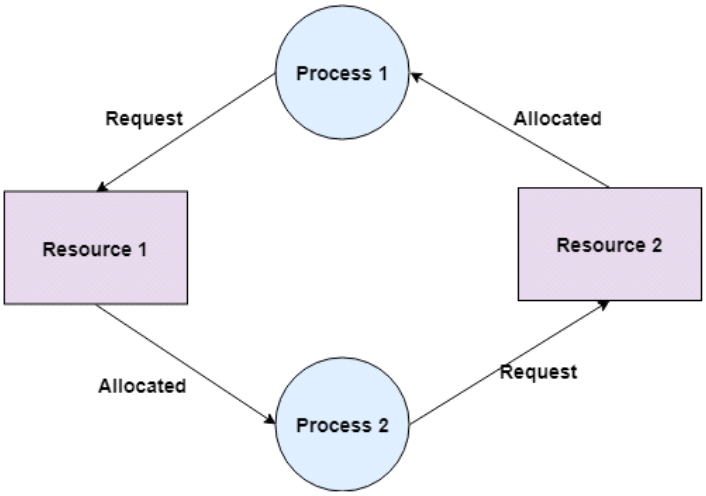
- **Hold and Wait**
  A process can hold multiple resources and still request more resources from other processes which are holding them. In the diagram given below, Process 2 holds Resource 2 and Resource 3 and is requesting the Resource 1 which is held by Process 1.



- **No Preemption**
  A resource cannot be preempted from a process by force. A process can only release a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1. It will only be released when Process 1 relinquishes it voluntarily after its execution is complete.



- **Circular Wait**
  A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



Deadlock Detection

A deadlock can be detected by a resource scheduler as it keeps track of all the resources that are allocated to different processes. After a deadlock is detected, it can be resolved using the following methods –

- All the processes that are involved in the deadlock are terminated. This is not a good approach as all the progress made by the processes is destroyed.
- Resources can be preempted from some processes and given to others till the deadlock is resolved.

Deadlock Prevention

It is very important to prevent a deadlock before it can occur. So, the system checks each transaction before it is executed to make sure it does not lead to deadlock. If there is even a slight chance that a transaction may lead to deadlock in the future, it is never allowed to execute.

Deadlock Avoidance

It is better to avoid a deadlock rather than take measures after the deadlock has occurred. The wait for graph can be used for deadlock avoidance. This is however only useful for smaller databases as it can get quite complex in larger databases.

From <https://www.tutorialspoint.com/process-deadlocks-in-operating-system>

What is fragmentation? Types of fragmentation.

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Internal Fragmentation

Memory block assigned to process is bigger. Some portion of memory is left unused, as it cannot be used by another process. The internal fragmentation can be reduced by effectively assigning the smallest partition but large enough for the process.

External Fragmentation

Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous, so it cannot be used. External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

Following are the important differences between Internal Fragmentation and External Fragmentation.

Sr. No.	Key	Internal Fragmentation	External Fragmentation
1	Definition	When there is a difference between required memory space vs allotted memory space, problem is termed as Internal Fragmentation.	When there are small and non-contiguous memory blocks which cannot be assigned to any process, the problem is termed as External Fragmentation.
2	Memory Block Size	Internal Fragmentation occurs when allotted memory blocks are of fixed size.	External Fragmentation occurs when allotted memory blocks are of varying size.
3	Occurrence	Internal Fragmentation occurs when a process needs more space than the size of allotted memory block or use less space.	External Fragmentation occurs when a process is removed from the main memory.
4	Solution	Best Fit Block Search is the solution for internal fragmentation.	Compaction is the solution for external fragmentation.
5	Process	Internal Fragmentation occurs when Paging is employed.	External Fragmentation occurs when Segmentation is employed.

From <https://www.tutorialspoint.com/difference-between-internal-fragmentation-and-external-fragmentation>

What exactly Spooling is all about?

- Difficulty Level : Easy
- Last Updated : 29 May, 2017

SPOOL is an acronym for **simultaneous peripheral operations on-line**. It is a kind of buffering mechanism or a process in which data is temporarily held to be used and executed by a device, program or the system. Data is sent to and stored in memory or other volatile storage until the program or computer requests it for execution.

In a computer system peripheral equipments, such as printers and punch card readers etc (batch processing), are very slow relative to the performance of the rest of the system. Getting input and output from the system was quickly seen to be a bottleneck. Here comes the need for spool.

Spooling works like a typical request queue where data, instructions and processes from multiple sources are accumulated for execution later on. Generally, it is maintained on computer's physical memory, buffers or the I/O device-specific interrupts. The spool is processed in FIFO manner i.e. whatever first instruction is there in the queue will be popped and executed.

Applications/Implementations of Spool:

1) The most common can be found in I/O devices like keyboard printers and mouse. For example, In printer, the documents/files that are sent to the printer are first stored in the memory or the printer spooler. Once the printer is ready, it fetches the data from the spool and prints it.

Even experienced a situation when suddenly for some seconds your mouse or keyboard stops working? Meanwhile, we usually click again and again here and there on the screen to check if its working or not. When it actually starts working, what and wherever we pressed during its hang state gets executed very fast because all the instructions got stored in the respective device's spool.

2) A batch processing system uses spooling to maintain a queue of ready-to-run jobs which can be started as soon as the system has the resources to process them.

3) Spooling is capable of overlapping I/O operation for one job with processor operations for another job. i.e. multiple processes can write documents to a print queue without waiting and resume with their work.

4) E-mail: an email is delivered by a MTA (Mail Transfer Agent) to a temporary storage area where it waits to be picked up by the MA (Mail User Agent)

5) Can also be used for generating Banner pages (these are the pages used in computerized printing in order to separate documents from each other and to identify e.g. the originator of the print request by username, an account number or a bin for pickup. Such pages are used in office environments where many people share the small number of available resources).

From <https://www.geeksforgeeks.org/what-exactly-spooling-is-all-about/>

What is semaphore and mutex (Differences might be asked)? Define Binary semaphore.

Mutex and Semaphore both provide synchronization services but they are not the same. Details about both Mutex and Semaphore are given below –

Mutex

Mutex is a mutual exclusion object that synchronizes access to a resource. It is created with a unique name at the start of a program. The Mutex is a locking mechanism that makes sure only one thread can acquire the Mutex at a time and enter the critical section. This thread only releases the Mutex when it exits the critical section.
This is shown with the help of the following example –

```
wait (mutex);
.....
Critical Section
.....
signal (mutex);
```

A Mutex is different than a semaphore as it is a locking mechanism while a semaphore is a signalling mechanism. A binary semaphore can be used as a Mutex but a Mutex can never be used as a semaphore.

Semaphore

A semaphore is a signalling mechanism and a thread that is waiting on a semaphore can be signaled by another thread. This is different than a mutex as the mutex can be signaled only by the thread that called the wait function.
A semaphore uses two atomic operations, wait and signal for process synchronization.
The wait operation decrements the value of its argument S, if it is positive. If S is negative or zero, then no operation is performed.

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

The signal operation increments the value of its argument S.

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

There are mainly two types of semaphores i.e. counting semaphores and binary semaphores.
Counting Semaphores are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources.
The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0.

From <https://www.tutorialspoint.com/mutex-vs-semaphore>

Belady's Anomaly in Page Replacement Algorithms

- Difficulty Level : Easy
- Last Updated : 14 Aug, 2019

Prerequisite – Page Replacement Algorithms

In Operating System, process data is loaded in fixed sized chunks and each chunk is referred to as a page. The processor loads these pages in the fixed sized chunks of memory called frames. Typically the size of each page is always equal to the frame size.

A page fault occurs when a page is not found in the memory, and needs to be loaded from the disk. If a page fault occurs and all memory frames have been already allocated, then replacement of a page in memory is required on the request of a new page. This is referred to as demand-paging. The choice of which page to replace is specified by a page replacement algorithms. The commonly used page replacement algorithms are FIFO, LRU, optimal page replacement algorithms etc.

Generally, on increasing the number of frames to a process' virtual memory, its execution becomes faster as less number of page faults occur. Sometimes the reverse happens, i.e. more number of page faults occur when more frames are allocated to a process. This most unexpected result is termed as **Belady's Anomaly**.

Bélády's anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.

This phenomenon is commonly experienced in following page replacement algorithms:

1. First in first out (FIFO)
2. Second chance algorithm
3. Random page replacement algorithm

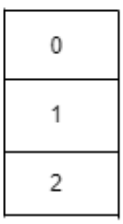
Reason of Belady's Anomaly –

The other two commonly used page replacement algorithms are Optimal and LRU, but Belady's Anamoly can never occur in these algorithms for any reference string as they belong to a class of stack based page replacement algorithms.

A **stack based algorithm** is one for which it can be shown that the set of pages in memory for *N* frames is always a subset of the set of pages that would be in memory with *N + 1* frames. For LRU replacement, the set of pages in memory would be the *n* most recently referenced pages. If the number of frames increases then these *n* pages will still be the most recently referenced and so, will still be in the memory. While in FIFO, if a page named *b* came into physical memory before

a page – *a* then priority of replacement of *b* is greater than that of *a*, but this is not independent of the number of page frames and hence, FIFO does not follow a stack page replacement policy and therefore suffers Belady's Anomaly.

Example: Consider the following diagram to understand the behaviour of a stack-based page replacement algorithm



Number of frames = 3 Number of frames = 4

As the set of pages in memory with 4 frames is not a subset of memory with 3 frames, the property of stack algorithm fails in FIFO.
The diagram illustrates that given the set of pages i.e. {0, 1, 2} in 3 frames of memory is not a subset of the pages in memory – {0, 1, 4, 5} with 4 frames and it is a violation in the property of stack based algorithms. This situation can be frequently seen in FIFO algorithm.

Belady's Anomaly in FIFO –

Assuming a system that has no pages loaded in the memory and uses the FIFO Page replacement algorithm. Consider the following reference string:

`1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5`

Case-1: If the system has 3 frames, the given reference string on using FIFO page replacement algorithm yields a total of 9 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

1	1	1	2	3	4	1	1	1	2	5	5
	2	2	3	4	1	2	2	2	5	3	3
		3	4	1	2	5	5	5	3	4	4
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	X

Case-2: If the system has 4 frames, the given reference string on using FIFO page replacement algorithm yields a total of 10 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

1	1	1	1	1	1	2	3	4	5	1	2
	2	2	2	2	2	3	4	5	1	2	3
		3	3	3	3	4	5	1	2	3	4
			4	4	4	5	1	2	3	4	5
PF	PF	PF	PF	X	X	PF	PF	PF	PF	PF	PF

It can be seen from the above example that on increasing the number of frames while using the FIFO page replacement algorithm, the number of **page faults increased** from 9 to 10.

Note – It is not necessary that every string reference pattern cause Belady anomaly in FIFO but there are certain kind of string references that worsen the FIFO performance on increasing the number of frames.

Why Stack based algorithms do not suffer Anomaly –

All the stack based algorithms never suffer Belady Anomaly because these type of algorithms assigns a priority to a page (for replacement) that is independent of the number of page frames. Examples of such policies are Optimal, LRU and LFU. Additionally these algorithms also have a good property for simulation, i.e. the miss (or hit) ratio can be computed for any number of page frames with a single pass through the reference string.

In LRU algorithm every time a page is referenced it is moved at the top of the stack, so, the top *n* pages of the stack are the *n* most recently used pages.Even if the number of frames are incremented to *n+1*, top of the stack will have *n+1* most recently used pages.

Similar example can be used to calculate the number of page faults in LRU algorithm. Assuming a system that has no pages loaded in the memory and uses the LRU Page replacement algorithm. Consider the following reference string:

`1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5`

Case-1: If the system has 3 frames, the given reference string on using LRU page replacement algorithm yields a total of 10 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.

1	2	3	4	1	2	5	1	2	3	4	5
	1	2	3	4	1	2	5	1	2	3	4
		1	2	3	4	1	2	5	1	2	3
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	PF

Case-2: If the system has 4 frames, the given reference string on using LRU page replacement algorithm, then total 8 page faults occur. The diagram shows the pattern of the page faults in the example.

1	2	3	4	1	2	5	1	2	3	4	5
	1	2	3	4	1	2	5	1	2	3	4
		1	2	3	4	1	2	5	1	2	3
			1	2	3	4	4	4	5	1	2
PF	PF	PF	PF	X	X	PF	X	X	PF	PF	PF

Conclusion –

Various factors substantially affect the number of page faults, such as reference string length and the number of free page frames available. Anomalies also occurs due to the small cache size as well as the reckless rate of change of the contents of cache. Also, the situation of fixed number of page faults even after increasing the number of frames can also be seen as an anomaly. Often algorithms like **Random page replacement algorithm** are also susceptible to Belady's Anomaly, because it **may behave like first in first out (FIFO)** page replacement algorithm. But Stack based algorithms are generally immune to all such situations as they are guaranteed to give better page hits when the frames are incremented.

From <https://www.geeksforgeeks.org/beladys-anomaly-in-page-replacement-algorithms/>

Starvation and Aging in Operating Systems

- Difficulty Level : Easy
- Last Updated : 16 Aug, 2019

Prerequisites : Priority Scheduling

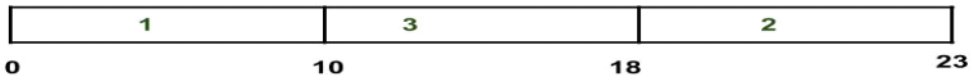
We have already discussed about the priority scheduling in this post. It is one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on.

In this post we will discuss a major problem related to priority scheduling and it's solution.

Starvation or indefinite blocking is phenomenon associated with the Priority scheduling algorithms, in which a process ready to run for CPU can wait indefinitely because of low priority. In heavily loaded computer system, a steady stream of higher-priority processes can prevent a low-priority process from ever getting the CPU.

There has been rumors that in 1967 Priority Scheduling was used in IBM 7094 at MIT , and they found a low-priority process that had not been submitted till 1973.

Process	Burst time	Priority
1	10	2
2	5	0
3	8	1



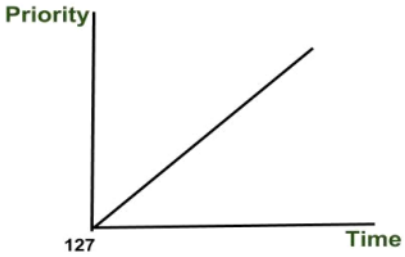
As we see in the above example process having higher priority than other processes getting CPU earlier. We can think of a scenario in which only one process is having very low-priority (for example 127) and we are giving other process with high-priority, this can lead indefinitely waiting for the process for CPU which is having low-priority, this leads to **Starvation**. Further we have also discuss about the solution of starvation.

Differences between Deadlock and Starvation in OS :

1. Deadlock occurs when none of the processes in the set is able to move ahead due to occupancy of the required resources by some other process as shown in the figure below, on the other hand Starvation occurs when a process waits for an indefinite period of time to get the resource it requires.
2. Other name of deadlock is **Circular Waiting**. Other name of starvation is **Lived lock**.
3. When deadlock occurs no process can make progress, while in starvation apart from the victim process other processes can progress or proceed.

Solution to Starvation : Aging

Aging is a technique of gradually increasing the priority of processes that wait in the system for a long time.For example, if priority range from 127(low) to 0(high), we could increase the priority of a waiting process by 1 Every 15 minutes. Eventually even a process with an initial priority of 127 would take no more than 32 hours for priority 127 process to age to a priority-0 process.



From <https://www.geeksforgeeks.org/starvation-and-aging-in-operating-systems/>

What is thrashing? Why does it occur?

In operating systems that implement a virtual memory space the programs allocate memory from an address space that may be much larger than the actual amount of RAM the system possesses. The OS is responsible for deciding which programs "memory" is in actual RAM. It needs a place to keep things while they are "out". This is what is called "swap space", as the OS is swapping things in and out as needed. When this swapping activity is occurring such that *it* is the major consumer of the CPU time, then you are effectively thrashing. You prevent it by running fewer programs, writing programs that use memory more efficiently, adding RAM to the system, or maybe even by increasing the swap size.
A page fault occurs when the memory access requested (from the virtual address space) does not map to something that is in RAM. A page must then be sent from RAM to swap, so that the requested new page can be brought from swap to RAM. As you might imagine, 2 disk I/Os for a RAM read tends to be pretty poor performance.

From <https://stackoverflow.com/questions/19031902/what-is-thrashing-why-does-it-occur>

What is paging and why do we need it?

Paging in Operating System

- Difficulty Level : Medium
- Last Updated : 28 Jun, 2021

Paging is a memory management scheme that eliminates the need for contiguous allocation of physical memory. This scheme permits the physical address space of a process to be non – contiguous.

- Logical Address or Virtual Address (represented in bits): An address generated by the CPU
- Logical Address Space or Virtual Address Space(represented in words or bytes): The set of all logical addresses generated by a program
- Physical Address (represented in bits): An address actually available on memory unit
- Physical Address Space (represented in words or bytes): The set of all physical addresses corresponding to the logical addresses

Example:

- If Logical Address = 31 bit, then Logical Address Space = 2^{31} words = 2 G words (1 G = 2^{30})
- If Logical Address Space = 128 M words = $2^7 * 2^{20}$ words, then Logical Address = $\log_2 2^{27}$ = 27 bits
- If Physical Address = 22 bit, then Physical Address Space = 2^{22} words = 4 M words (1 M = 2^{20})
- If Physical Address Space = 16 M words = $2^4 * 2^{20}$ words, then Physical Address = $\log_2 2^{24}$ = 24 bits

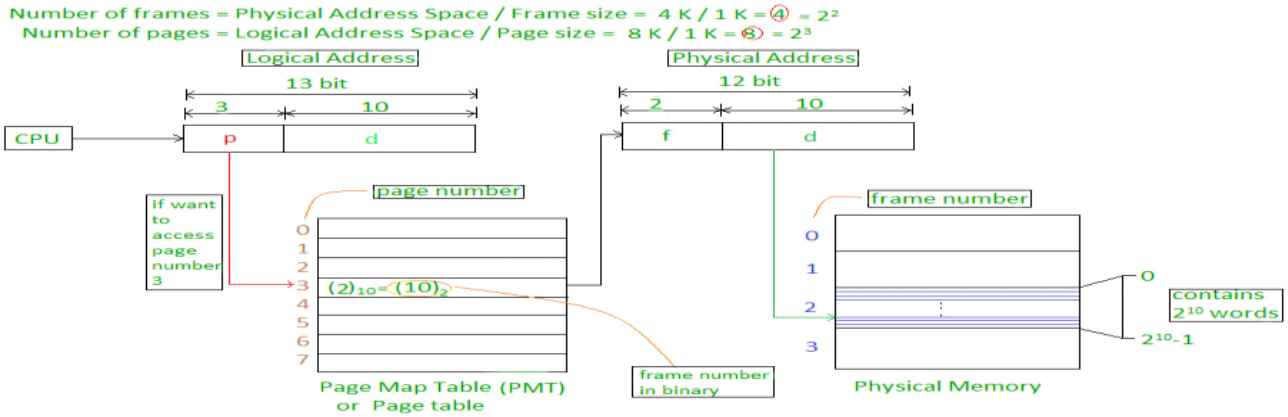
The mapping from virtual to physical address is done by the memory management unit (MMU) which is a hardware device and this mapping is known as paging technique.

- The Physical Address Space is conceptually divided into a number of fixed-size blocks, called **frames**.
- The Logical address Space is also splitted into fixed-size blocks, called **pages**.
- Page Size = Frame Size

Let us consider an example:

- Physical Address = 12 bits, then Physical Address Space = 4 K words
- Logical Address = 13 bits, then Logical Address Space = 8 K words

- Page size = frame size = 1 K words (assumption)

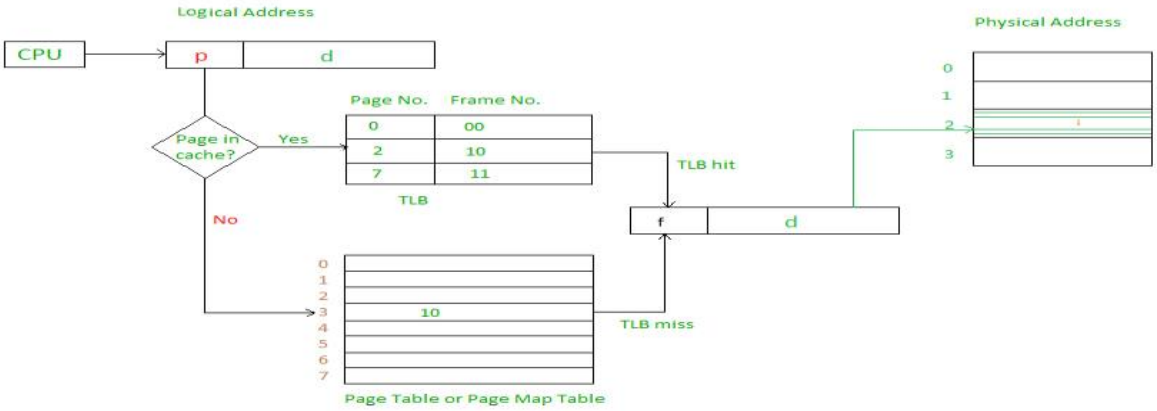


Address generated by CPU is divided into

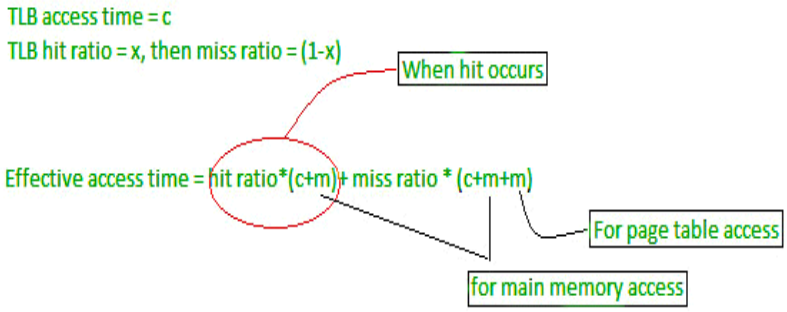
- **Page number(p):** Number of bits required to represent the pages in Logical Address Space or Page number
- **Page offset(d):** Number of bits required to represent particular word in a page or page size of Logical Address Space or word number of a page or page offset. Physical Address is divided into
- **Frame number(f):** Number of bits required to represent the frame of Physical Address Space or Frame number.
- **Frame offset(d):** Number of bits required to represent particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

The hardware implementation of page table can be done by using dedicated registers. But the usage of register for the page table is satisfactory only if page table is small. If page table contain large number of entries then we can use TLB(translation Look-aside buffer), a special, small, fast look up hardware cache.

- The TLB is associative, high speed memory.
- Each entry in TLB consists of two parts: a tag and a value.
- When this memory is used, then an item is compared with all tags simultaneously.If the item is found, then corresponding value is returned.



Main memory access time = m
If page table are kept in main memory,
Effective access time = m(for page table) + m(for particular page in page table)



From <https://www.geeksforgeeks.org/paging-in-operating-system/>

Difference between Demand Paging and Segmentation

- Difficulty Level : Easy
- Last Updated : 02 Mar, 2020

Demand Paging:

Demand paging is identical to the paging system with swapping. In demand paging, a page is delivered into the memory on demand i.e., only when a reference is made to a location on that page. Demand paging combines the feature of simple paging and implement virtual memory as it has a large virtual memory. Lazy swapper concept is implemented in demand paging in which a page is not swapped into the memory unless it is required.

Segmentation:

Segmentation is the arrangement of memory management. According to the segmentation the logical address space is a collection of segments. Each segment has a name and length. Each logical address have two quantities segment name and the segment offset, for simplicity we use the segment number in place of segment name.

The difference between Demand Paging and Segmentation are as follows:

S.No.	Demand Paging	Segmentation
1.	In demand paging, the pages are of equal size.	While in segmentation, segments can be of different size.
2.	Page size is fixed in the demand paging.	Segment size may vary in segmentation as it grants dynamic increase of segments.
3.	It does not allows sharing of the pages.	While segments can be shared in segmentation.
4.	In demand paging, on demand pages are loaded in the memory.	In segmentation, during compilation segments are allocated to the program.
5.	Page map table in demand paging manages record of pages in memory.	Segment map table in segmentation demonstrates every segment address in the memory.
6.	It provides large virtual memory and have more efficient use of memory.	It provides virtual memory and maximum size of segment is defined by the size of memory.

From <https://www.geeksforgeeks.org/difference-between-demand-paging-and-segmentation/>

Real Time Operating System, types of RTOS

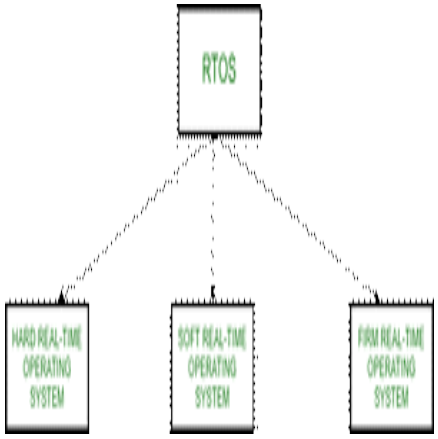
Real Time Operating System (RTOS)

- Difficulty Level : Medium
- Last Updated : 02 Jul, 2021

Real-time **operating systems (RTOS)** are used in environments where a large number of events, mostly external to the computer system, must be accepted and processed in a short time or within certain deadlines. such applications are industrial control, telephone switching equipment, flight control, and real-time simulations. With an RTOS, the processing time is measured in tenths of seconds. This system is time-bound and has a fixed deadline. The processing in this type of system must occur within the specified constraints. Otherwise, This will lead to system failure.

Examples of the real-time operating systems: Airline traffic control systems, Command Control Systems, Airlines reservation system, Heart Peacemaker, Network Multimedia Systems, Robot etc.

The real-time operating systems can be of 3 types –



1. **Hard** Real-Time **operating system:**
   These operating systems guarantee that critical tasks be completed within a range of time.
   For example, a robot is hired to weld a car body. If the robot welds too early or too late, the car cannot be sold, so it is a hard real-time system that requires complete car welding by robot hardly on the time.

2. **Soft** real-time **operating system:**
   This operating system provides some relaxation in the time limit.
   For example – Multimedia systems, digital audio systems etc. Explicit, programmer-defined and controlled processes are encountered in real-time systems. A separate process is changed with handling a single external event. The process is activated upon occurrence of the related event signalled by an interrupt.

   Multitasking operation is accomplished by scheduling processes for execution independently of each other. Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services. The processor is allocated to the highest priority processes. This type of schedule, called, priority-based preemptive scheduling is used by real-time systems.

3. **Firm** Real-time **Operating System**:
   RTOS of this type have to follow deadlines as well. In spite of its small impact, missing a deadline can have unintended consequences, including a reduction in the quality of the product. Example: Multimedia applications.

   Advantages:

   The advantages of real-time operating systems are as follows-

1. **Maximum consumption –**
   Maximum utilization of devices and systems. Thus more output from all the resources.

2. **Task Shifting –**
   Time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds. Shifting one task to another and in the latest systems, it takes 3 microseconds.

3. **Focus On Application –**
   Focus on running applications and less importance to applications that are in the queue.

4. Real-Time **Operating System In Embedded System –**
   Since the size of programs is small, RTOS can also be embedded systems like in transport and others.

5. **Error Free –**
   These types of systems are error-free.

6. **Memory Allocation –**
   Memory allocation is best managed in these types of systems.
   Disadvantages:

   The disadvantages of real-time operating systems are as follows-

1. **Limited Tasks –**
   Very few tasks run simultaneously, and their concentration is very less on few applications to avoid errors.

2. **Use Heavy System Resources –**
   Sometimes the system resources are not so good and they are expensive as well.

3. **Complex Algorithms –**
   The algorithms are very complex and difficult for the designer to write on.

4. **Device Driver And Interrupt signals –**
   It needs specific device drivers and interrupts signals to respond earliest to interrupts.

5. **Thread Priority –**
   It is not good to set thread priority as these systems are very less prone to switching tasks.

6. **Minimum Switching –** RTOS performs minimal task switching.

From <https://www.geeksforgeeks.org/real-time-operating-system-rtos/>

Difference between main memory and secondary memory.

A memory is just like a human brain. It is used to store data and instructions. Computer memory is the storage space in the computer, where data is to be processed and instructions required for processing are stored. The memory is divided into large number of small parts called cells. Each location or cell has a unique address, which varies from zero to memory size minus one. For example, if the computer has 64k words, then this memory unit has 64 * 1024 = 65536 memory locations. The address of these locations varies from 0 to 65535.
Memory is primarily of three types –

- Cache Memory
- Primary Memory/Main Memory
- Secondary Memory

Cache Memory

Cache memory is a very high speed semiconductor memory which can speed up the CPU. It acts as a buffer between the CPU and the main memory. It is used to hold those parts of data and program which are most frequently used by the CPU. The parts of data and programs are transferred from the disk to cache memory by the operating system, from where the CPU can access them.

Advantages

The advantages of cache memory are as follows –

- Cache memory is faster than main memory.
- It consumes less access time as compared to main memory.

- It stores the program that can be executed within a short period of time.
- It stores data for temporary use.

Disadvantages

The disadvantages of cache memory are as follows –

- Cache memory has limited capacity.
- It is very expensive.

Primary Memory (Main Memory)

Primary memory holds only those data and instructions on which the computer is currently working. It has a limited capacity and data is lost when power is switched off. It is generally made up of semiconductor device. These memories are not as fast as registers. The data and instruction required to be processed resides in the main memory. It is divided into two subcategories RAM and ROM.

Characteristics of Main Memory

- These are semiconductor memories.
- It is known as the main memory.
- Usually volatile memory.
- Data is lost in case power is switched off.
- It is the working memory of the computer.
- Faster than secondary memories.
- A computer cannot run without the primary memory.

Secondary Memory

This type of memory is also known as external memory or non-volatile. It is slower than the main memory. These are used for storing data/information permanently. CPU directly does not access these memories, instead they are accessed via input-output routines. The contents of secondary memories are first transferred to the main memory, and then the CPU can access it. For example, disk, CD-ROM, DVD, etc.

Characteristics of Secondary Memory

- These are magnetic and optical memories.
- It is known as the backup memory.
- It is a non-volatile memory.
- Data is permanently stored even if power is switched off.
- It is used for storage of data in a computer.
- Computer may run without the secondary memory.
- Slower than primary memories.

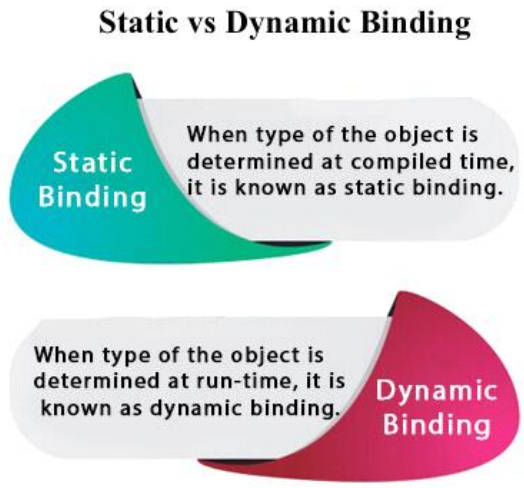
From <https://www.tutorialspoint.com/computer_fundamentals/computer_memory.htm>

Static Binding and Dynamic Binding

Connecting a method call to the method body is known as binding.

There are two types of binding

1. Static Binding (also known as Early Binding).
2. Dynamic Binding (also known as Late Binding).



From <https://www.javatpoint.com/static-binding-and-dynamic-binding>

Operating System Scheduling algorithms

Advertisements
Previous Page
Next Page
A Process Scheduler schedules different processes to be assigned to the CPU based on particular scheduling algorithms. There are six popular process scheduling algorithms which we are going to discuss in this chapter –

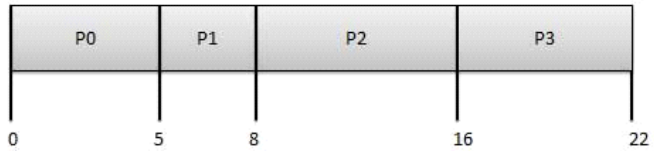
- First-Come, First-Served (FCFS) Scheduling
- Shortest-Job-Next (SJN) Scheduling
- Priority Scheduling
- Shortest Remaining Time
- Round Robin(RR) Scheduling
- Multiple-Level Queues Scheduling

These algorithms are either non-preemptive or preemptive. Non-preemptive algorithms are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the preemptive scheduling is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters into a ready state.

First Come First Serve (FCFS)

- Jobs are executed on first come, first serve basis.
- It is a non-preemptive, pre-emptive scheduling algorithm.
- Easy to understand and implement.
- Its implementation is based on FIFO queue.
- Poor in performance as average wait time is high.

Process	Arrival Time	Execute Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	0 - 0 = 0

P1	5 - 1 = 4
P2	8 - 2 = 6
P3	16 - 3 = 13

Average Wait Time: (0+4+6+13) / 4 = 5.75

Shortest Job Next (SJN)

- This is also known as shortest job first, or SJF
- This is a non-preemptive, pre-emptive scheduling algorithm.
- Best approach to minimize waiting time.
- Easy to implement in Batch systems where required CPU time is known in advance.
- Impossible to implement in interactive systems where required CPU time is not known.
- The processer should know in advance how much time process will take.

Given: Table of processes, and their Arrival time, Execution time

Process	Arrival Time	Execution Time	Service Time
P0	0	5	0
P1	1	3	5
P2	2	8	14
P3	3	6	8

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	5 - 1 = 4
P2	14 - 2 = 12
P3	8 - 3 = 5

Average Wait Time: (0 + 4 + 12 + 5)/4 = 21 / 4 = 5.25

Priority Based Scheduling

- Priority scheduling is a non-preemptive algorithm and one of the most common scheduling algorithms in batch systems.
- Each process is assigned a priority. Process with highest priority is to be executed first and so on.
- Processes with same priority are executed on first come first served basis.
- Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Given: Table of processes, and their Arrival time, Execution time, and priority. Here we are considering 1 is the lowest priority.

Process	Arrival Time	Execution Time	Priority	Service Time
P0	0	5	1	0
P1	1	3	2	11
P2	2	8	1	14
P3	3	6	3	5

Waiting time of each process is as follows –

Process	Waiting Time
P0	0 - 0 = 0
P1	11 - 1 = 10
P2	14 - 2 = 12
P3	5 - 3 = 2

Average Wait Time: (0 + 10 + 12 + 2)/4 = 24 / 4 = 6

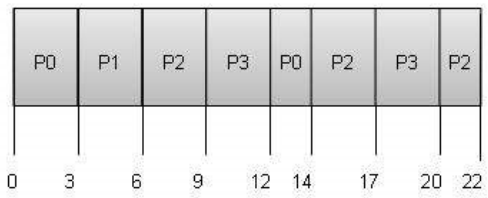
Shortest Remaining Time

- Shortest remaining time (SRT) is the preemptive version of the SJN algorithm.
- The processor is allocated to the job closest to completion but it can be preempted by a newer ready job with shorter time to completion.
- Impossible to implement in interactive systems where required CPU time is not known.
- It is often used in batch environments where short jobs need to give preference.

Round Robin Scheduling

- Round Robin is the preemptive process scheduling algorithm.
- Each process is provided a fix time to execute, it is called a quantum.
- Once a process is executed for a given time period, it is preempted and other process executes for a given time period.
- Context switching is used to save states of preempted processes.

Quantum = 3



Wait time of each process is as follows –

Process	Wait Time : Service Time - Arrival Time
P0	(0 - 0) + (12 - 3) = 9
P1	(3 - 1) = 2
P2	(6 - 2) + (14 - 9) + (20 - 17) = 12
P3	(9 - 3) + (17 - 12) = 11

Average Wait Time: (9+2+12+11) / 4 = 8.5

Multiple-Level Queues Scheduling

Multiple-level queues are not an independent scheduling algorithm. They make use of other existing algorithms to group and schedule jobs with common characteristics.

- Multiple queues are maintained for processes with common characteristics.
- Each queue can have its own scheduling algorithms.
- Priorities are assigned to each queue.

For example, CPU-bound jobs can be scheduled in one queue and all I/O-bound jobs in another queue. The Process Scheduler then alternately selects jobs from each queue and assigns them to the CPU based on the algorithm assigned to the queue.

From <https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm>

Producer Consumer Problem using Semaphores

Computer ScienceMCAOperating System

The producer consumer problem is a synchronization problem. There is a fixed size buffer and the producer produces items and enters them into the buffer. The consumer removes the items from the buffer and consumes them.
A producer should not produce items into the buffer when the consumer is consuming an item from the buffer and vice versa. So the buffer should only be accessed by the producer or consumer at a time.
The producer consumer problem can be resolved using semaphores. The codes for the producer and consumer process are given as follows –

Producer Process

The code that defines the producer process is given below –

```
do {
   .
   . PRODUCE ITEM
   .
   wait(empty);
   wait(mutex);
   .
   . PUT ITEM IN BUFFER
   .
   signal(mutex);
   signal(full);

} while(1);
```

In the above code, mutex, empty and full are semaphores. Here mutex is initialized to 1, empty is initialized to n (maximum size of the buffer) and full is initialized to 0.
The mutex semaphore ensures mutual exclusion. The empty and full semaphores count the number of empty and full spaces in the buffer.
After the item is produced, wait operation is carried out on empty. This indicates that the empty space in the buffer has decreased by 1. Then wait operation is carried out on mutex so that consumer process cannot interfere.
After the item is put in the buffer, signal operation is carried out on mutex and full. The former indicates that consumer process can now act and the latter shows that the buffer is full by 1.

Consumer Process

The code that defines the consumer process is given below:

```
do {

   wait(full);
   wait(mutex);
   . .
   . REMOVE ITEM FROM BUFFER
   .
   signal(mutex);
   signal(empty);
   .
   . CONSUME ITEM
   .
} while(1);
```

The wait operation is carried out on full. This indicates that items in the buffer have decreased by 1. Then wait operation is carried out on mutex so that producer process cannot interfere.

Then the item is removed from buffer. After that, signal operation is carried out on mutex and empty. The former indicates that consumer process can now act and the latter shows that the empty space in the buffer has increased by 1.

From <https://www.tutorialspoint.com/producer-consumer-problem-using-semaphores>

Banker's Algorithm in Operating System [Example]

What is Banker's Algorithm?

Banker's Algorithm is used majorly in the banking system to avoid deadlock. It helps you to identify whether a loan will be given or not.
This algorithm is used to test for safely simulating the allocation for determining the maximum amount available for all resources. It also checks for all the possible activities before determining whether allocation should be continued or not.
For example, there are X number of account holders of a specific bank, and the total amount of money of their accounts is G.
When the bank processes a car loan, the software system subtracts the amount of loan granted for purchasing a car from the total money (G+ Fixed deposit + Monthly Income Scheme + Gold, etc.) that the bank has.
35.5M
358
It also checks that the difference is more than or not G. It only processes the car loan when the bank has sufficient money even if all account holders withdraw the money G simultaneously.
In this operating system tutorial, you will learn:

- What is Banker's Algorithm?
- Banker's Algorithm Notations
- Example of Banker's algorithm
- Characteristics of Banker's Algorithm
- Disadvantage of Banker's algorithm

Banker's Algorithm Notations

Here is an important notation used in Banker's algorithm:

- X: Indicates the total number of processes of the system.
- Y: Indicates the total number of resources present in the system.

Available

[I: Y] indicate which resource is available.

Max

[I:X,I: Y]: Expression of the maximum number of resources of type j or process i

Allocation

[I:X,I:Y]. Indicate where process you have received a resource of type j

Need

Express how many more resources can be allocated in the future

Example of Banker's algorithm

Assume that we have the following resources:

- 5 Pen drives
- 2 Printers
- 4 Scanners
- 3 Hard disks

Here, we have created a vector representing total resources: Available = (5, 2, 4, 3).
Assume there are four processes. The available resources are already allocated as per the matrix table below.

Process Name	Pen Drives	Printer	Scanner	Hard disk
P	2	0	1	1
Q	0	1	0	0
R	1	0	1	1
S	1	1	0	1
Total	4	2	2	3

Here, the allocated resources is the total of these columns:
Allocated = (4, 2, 2, 3).
We also create a Matrix to display the number of each resource required for all the processes. This matrix is called **Need**=(3,0,2,2)

Process Name	Pen Drives	Printer	Scanner	Hard disk

P	1	1	0	0
Q	0	1	1	2
R	2	1	0	0
S	0	0	1	0

The available vector will be :
Available=Available- Allocated
= (5, 2, 4, 3) -(4, 2, 2, 3)
=(1, 0, 2, 0)

Resource Request Algorithm

Resource request algorithm enables you to represent the system behavior when a specific process makes a resource request.
Let understand this by the following steps:
Step 1) When a total requested instance of all resources is lesser than the process, move to step 2.
Step 2) When a requested instance of each and every resource type is lesser compared to the available resources of each type, it will be processed to the next step. Otherwise, the process requires to wait because of the unavailability of sufficient resources.
Step 3) Resource is allocated as shown in the below given Pseudocode.

```
Available = Available – Request (y)
Allocation(x) = Allocation(x) + Request(x)
Need(x) = Need(x) - Request(x)
```

This final step is performed because the system needs to assume that resources have been allocated. So that there should be less resources available after allocation.

Characteristics of Banker's Algorithm

Here are important characteristics of banker's algorithm:

- Keep many resources that satisfy the requirement of at least one client
- Whenever a process gets all its resources, it needs to return them in a restricted period.
- When a process requests a resource, it needs to wait
- The system has a limited number of resources
- Advance feature for max resource allocation

Disadvantage of Banker's algorithm

Here, are cons/drawbacks of using banker's algorithm

- Does not allow the process to change its Maximum need while processing
- It allows all requests to be granted in restricted time, but one year is a fixed period for that.
- All processes must know and state their maximum resource needs in advance.

Summary:

- Banker's algorithm is used majorly in the banking system to avoid deadlock. It helps you to identify whether a loan will be given or not.
- Notations used in banker's algorithms are 1) Available 2) Max 3) Allocation 4) Need
- Resource request algorithm enables you to represent the system behavior when a specific process makes a resource request.
- Banker's algorithm keeps many resources that satisfy the requirement of at least one client
- The biggest drawback of banker's algorithm this that it does not allow the process to change its Maximum need while processing.

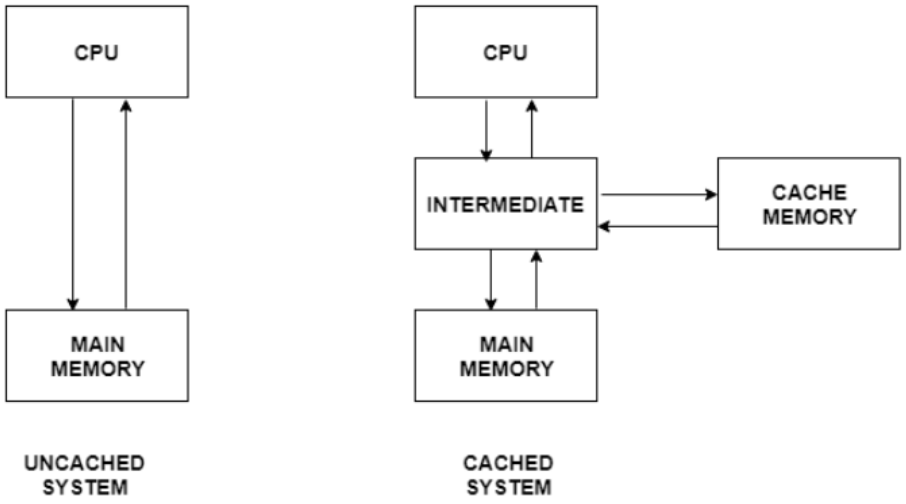
From <https://www.guru99.com/bankers-algorithm-in-operating-system.html>

Explain Cache

Cache is a type of memory that is used to increase the speed of data access. Normally, the data required for any process resides in the main memory. However, it is transferred to the cache memory temporarily if it is used frequently enough. The process of storing and accessing data from a cache is known as caching.

Uncached System vs Cached System

A figure to better understand the difference between cached and uncached system is as follows –



Some important points to explain the above figureare –

- In an uncached system, there is no cache memory. So, all the data required by the processor during execution is obtained from main memory. This is a comparatively time consuming process.
- In contrast to this, a cached system contains a cache memory. Any data required by the processor is searched in the cache memory first. If it is not available there then main memory is searched. The cache system yields faster results than the uncached system because cache is much faster than main memory.

Advantages of Cache Memory

Some of the advantages of cache memory are as follows –

- Cache memory is faster than main memory as it is located on the processor chip itself. Its speed is comparable to the processor registers and so frequently required data is stored in the cache memory.
- The memory access time is considerably less for cache memory as it is quite fast. This leads to faster execution of any process.
- The cache memory can store data temporarily as long as it is frequently required. After the use of any data has ended, it can be removed from the cache and replaced by new data from the main memory.

Disadvantages of Cache Memory

Some of the disadvantages of cache memory are as follows –

- Since the cache memory is quite fast, it is extremely useful in any computer system. However, it is also quite expensive and so is used judiciously.
- The cache is memory expensive as observed from the previous point. Also, it is located directly on the processor chip. Because of these reasons, it has a limited capacity and is much smaller than main memory.

From <https://www.tutorialspoint.com/What-is-caching>

Diff between direct mapping and associative mapping

In a cache system, **direct mapping** maps each block of main memory into only one possible cache line.
Associative mapping permits each main memory block to be loaded into any line of the cache.
In **set-associative mapping**, the cache is divided into a number of sets of cache lines; each main memory block can be mapped into any line in a particular set.

From <https://cpentalk.com/687/differences-mapping-associative-mapping-associative-mapping>

Difference Between Multiprogramming and Multitasking

ProgrammingProgramming ScriptsSoftware & Coding

In this post, we will understand the difference between multiprogramming and

multitasking.

	Multiprogramming	Multitasking
What it is?	The concurrent residency of more than one program in the main memory is called as multiprogramming.	The execution of more than one task simultaneously is called as multitasking.
Number of CPU	One	One
Job processing time:	More time is taken to process the jobs.	Moderate amount of time.
Number of process being executed	One process is executed at a time.	One by one job is being executed at a time.
Economical	It is economical.	It is economical.
Number of users:	One at a time.	More than one.
Throughput	Throughput is less.	Throughput is moderate.
Efficiency	Less	Moderate
Categories	No further divisions	Single User & Multiuser.

From <https://www.tutorialspoint.com/difference-between-multiprogramming-and-multitasking>

The End