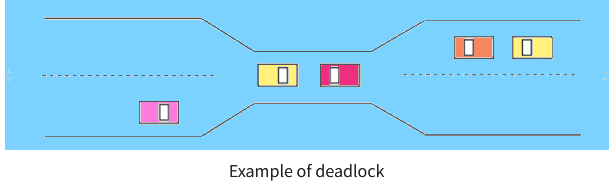
**What is Deadlock?**

**Deadlock** is a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software.

## Example of Deadlock

* A real-world example would be traffic, which is going only in one direction.
* Here, a bridge is considered a resource.
* So, when Deadlock happens, it can be easily resolved if one car backs up (Preempt resources and rollback).
* Several cars may have to be backed up if a deadlock situation occurs.
* So starvation is possible.



## Methods of deadlock handling

## Deadlock Prevention in OS

It’s important to prevent a deadlock before it can occur. The system checks every transaction before it is executed to make sure it doesn’t lead the deadlock situations. Such that even a small change to occur dead that an operation which can lead to Deadlock in the future it also never allowed process to execute.

It is a set of methods for ensuring that at least one of the conditions cannot hold.

## Deadlock Detection in OS

A deadlock occurrence can be detected by the resource scheduler. A resource scheduler helps OS to keep track of all the resources which are allocated to different processes. So, when a deadlock is detected, it can be resolved using the below-given methods

## Deadlock Avoidance Algorithms

It is better to avoid a deadlock instead of taking action after the Deadlock has occurred. It needs additional information, like how resources should be used. Deadlock avoidance is the simplest and most useful model that each process declares the maximum number of resources of each type that it may need.

**What is Memory Management?**  
Memory Management is the process of controlling and coordinating  
computer memory, assigning portions known as blocks to various  
running programs to optimize the overall performance of the system.  
It is the most important function of an operating system that manages  
primary memory. It helps processes to move back and forward  
between the main memory and execution disk. It helps OS to keep  
track of every memory location, irrespective of whether it is allocated  
to some process or it remains free

**Swapping:**  
Swapping is a mechanism in which a process can  
be swapped temporarily out of main memory (or move) to secondary  
storage (disk) and make that memory available to other processes. At  
some later time, the system swaps back the process from the secondary  
storage to main memory

**What is Memory allocation?**  
Memory allocation is a process by which computer programs are  
assigned memory or space.  
Here, main memory is divided into two types of partitions  
1. Low Memory - Operating system resides in this type of  
memory.  
2. High Memory- User processes are held in high memory.  
**Partition Allocation**  
Memory is divided into different blocks or partitions. Each process is  
allocated according to the requirement. Partition allocation is an ideal  
method to avoid internal fragmentation.  
Below are the various partition allocation schemes :  
• First Fit: In this type fit, the partition is allocated, which is the  
first sufficient block from the beginning of the main memory.  
• Best Fit: It allocates the process to the partition that is the first  
smallest partition among the free partitions.  
• Worst Fit: It allocates the process to the partition, which is the  
largest sufficient freely available partition in the main memory.  
• Next Fit: It is mostly similar to the first Fit, but this Fit, searches  
for the first sufficient partition from the last allocation point.

**What is Segmentation?**  
Segmentation method works almost similarly to paging. The only  
difference between the two is that segments are of variable-length,  
whereas, in the paging method, pages are always of fixed size.  
A program segment includes the program's main function, data  
structures, utility functions, etc. The OS maintains a segment map  
table for all the processes. It also includes a list of free memory blocks  
along with its size, segment numbers, and its memory locations in the  
main memory or virtual memory.

**What is Paging?**  
Paging is a storage mechanism that allows OS to retrieve processes  
from the secondary storage into the main memory in the form of  
pages. In the Paging method, the main memory is divided into small  
fixed-size blocks of physical memory, which is called frames. The  
size of a frame should be kept the same as that of a page to have  
maximum utilization of the main memory and to avoid external  
fragmentation. Paging is used for faster access to data, and it is a  
logical concept.

**Concepts of Virtual memory**

Virtual memory is a feature of an operating system that enables a computer to be able to compensate shortages of physical memory by transferring pages of data from random access memory to disk storage.

This process is done temporarily and is designed to work as a combination of RAM and space on the hard disk.

This means that when RAM runs low, virtual memory can move data from it to a space called a paging file. This process allows for RAM to be freed up so that a computer can complete the task.

**Demand Paging**

Demand paging is a type of swapping done in virtual memory systems. In demand paging, the data is not copied from the disk to the RAM until they are needed or being demanded by some program. The data will not be copied when the data is already available on the memory. This is otherwise called a lazy evaluation because only the demanded pages of memory are being swapped from the secondary storage (disk space) to the main memory. In contrast during pure swapping, all the memory for a process is swapped from secondary storage to main memory during the process startup.

Occasionally a user might be shown a message that says the virtual memory is running low, this means that either more RAM needs to be added, or the size of the paging file needs to be increased.

# Page Replacement Algorithms in Operating Systems

* Difficulty Level : [Easy](https://www.geeksforgeeks.org/easy/)
* Last Updated : 04 Aug, 2021

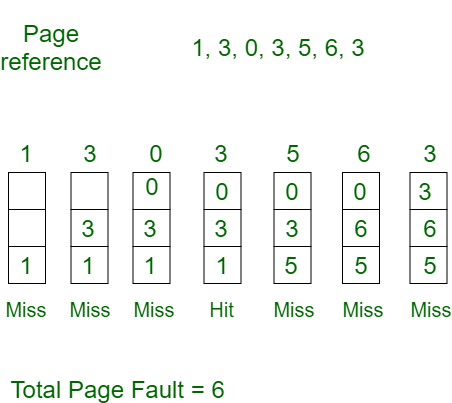
In an operating system that uses paging for memory management, a page replacement algorithm is needed to decide which page needs to be replaced when new page comes in.

**Page Fault –** A page fault happens when a running program accesses a memory page that is mapped into the virtual address space, but not loaded in physical memory.

Since actual physical memory is much smaller than virtual memory, page faults happen. In case of page fault, Operating System might have to replace one of the existing pages with the newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce the number of page faults.

**Page Replacement Algorithms :**

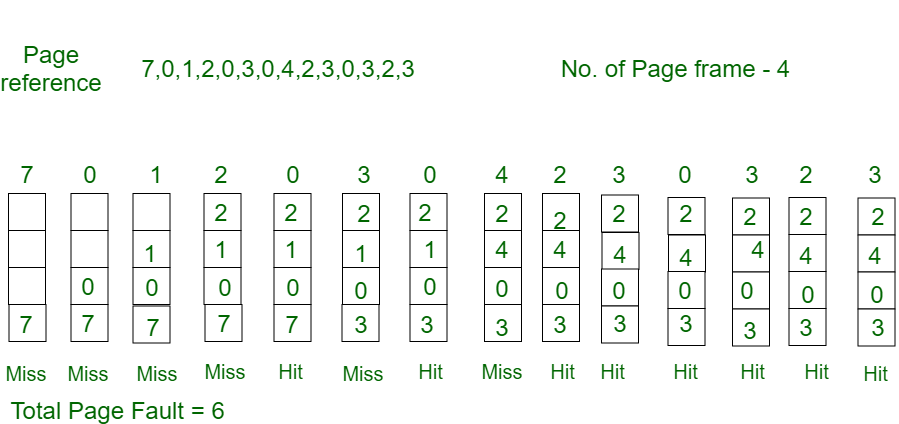
**1. First In First Out (FIFO) –**   
This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.   
**Example-1**Consider page reference string 1, 3, 0, 3, 5, 6 with 3 page frames.Find number of page faults.



Initially all slots are empty, so when 1, 3, 0 came they are allocated to the empty slots —> **3 Page Faults.**   
when 3 comes, it is already in  memory so —> **0 Page Faults.**   
Then 5 comes, it is not available in  memory so it replaces the oldest page slot i.e 1. —>**1 Page Fault.**   
6 comes, it is also not available in memory so it replaces the oldest page slot i.e 3 —>**1 Page Fault.**   
Finally when 3 come it is not available so it replaces 0 **1 page fault**

[**Belady’s anomaly**](https://www.geeksforgeeks.org/operating-system-beladys-anomaly/) **–** Belady’s anomaly proves that it is possible to have more page faults when increasing the number of page frames while using the First in First Out (FIFO) page replacement algorithm.  For example, if we consider reference string 3, 2, 1, 0, 3, 2, 4, 3, 2, 1, 0, 4 and 3 slots, we get 9 total page faults, but if we increase slots to 4, we get 10 page faults.

**2. Optimal Page replacement –**   
In this algorithm, pages are replaced which would not be used for the longest duration of time in the future.   
**Example-2:**Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, with 4 page frame. Find number of page fault.



Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**   
0 is already there so —> **0 Page fault.**   
when 3 came it will take the place of 7 because it is not used for the longest duration of time in the future.—>**1 Page fault.**   
0 is already there so —> **0 Page fault.**.   
4 will takes place of 1 —> **1 Page Fault.**

Now for the further page reference string —> **0 Page fault** because they are already available in the memory.   
Optimal page replacement is perfect, but not possible in practice as the operating system cannot know future requests. The use of Optimal Page replacement is to set up a benchmark so that other replacement algorithms can be analyzed against it.

**3. Least Recently Used –**   
In this algorithm page will be replaced which is least recently used.   
**Example-3**Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2 with 4 page frames.Find number of page faults.

Initially all slots are empty, so when 7 0 1 2 are allocated to the empty slots —> **4 Page faults**   
0 is already their so —> **0 Page fault.**   
when 3 came it will take the place of 7 because it is least recently used —>**1 Page fault**   
0 is already in memory so —> **0 Page fault**.   
4 will takes place of 1 —> **1 Page Fault**   
Now for the further page reference string —> **0 Page fault** because they are already available in the memory.

# Disk Scheduling

As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.

However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.

The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

### Seek Time

Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.

### Rotational Latency

It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.

### Transfer Time

It is the time taken to transfer the data.

### Disk Access Time

Disk access time is given as,

Disk Access Time = Rotational Latency + Seek Time + Transfer Time

### Disk Response Time

It is the average of time spent by each request waiting for the IO operation.

### Purpose of Disk Scheduling

The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

### Goal of Disk Scheduling Algorithm

* Fairness
* High throughout
* Minimal traveling head time