

Basic Concepts

Virtual Memory

* An imaginary [memory](https://www.webopedia.com/TERM/M/memory.html) area [supported](https://www.webopedia.com/TERM/S/support.html) by some [operating systems](https://www.webopedia.com/TERM/O/operating_system.html) in conjunction with the [hardware](https://www.webopedia.com/TERM/H/hardware.html)
* [Programs](https://www.webopedia.com/TERM/P/program.html) use these *virtual addresses* rather than real addresses to [store](https://www.webopedia.com/TERM/S/store.html) [instructions](https://www.webopedia.com/TERM/I/instruction.html) and [data](https://www.webopedia.com/TERM/D/data.html). When the program is actually [executed](https://www.webopedia.com/TERM/E/execute.html), the virtual addresses are [converted](https://www.webopedia.com/TERM/C/convert.html) into real memory addresses.
* The purpose of virtual memory is to enlarge the [*address space*](https://www.webopedia.com/TERM/A/address_space.html), the set of addresses a program can utilize. For example, virtual memory might contain twice as many addresses as [main memory](https://www.webopedia.com/TERM/M/main_memory.html)

Segmentation

* In Operating Systems, Segmentation is a memory management technique in which, the memory is divided into the variable size parts. Each part is known as segment which can be allocated to a process.
* The details about each segment are stored in a table called as segment table. Segment table is stored in one (or many) of the segments.
* Segment table contains mainly two information about segment :-

1 Base: It is the base address of the segment

2 Limit: It is the length of the segment

Page Replacement Algorithm

* **page replacement algorithms** decide which memory pages to page out, sometimes called swap out, or write to disk, when a [page](https://en.wikipedia.org/wiki/Page_(computer_memory)) of memory needs to be allocated.
* [Page replacement](https://en.wikipedia.org/wiki/Paging) happens when a requested page is not in memory ([page fault](https://en.wikipedia.org/wiki/Page_fault)) and a free page cannot be used to satisfy the allocation, either because there are none, or because the number of free pages is lower than some threshold.

Paging

* **paging** is a [memory management](https://en.wikipedia.org/wiki/Memory_management) scheme by which a computer stores and retrieves data from [secondary storage](https://en.wikipedia.org/wiki/Computer_data_storage#Secondary_storage)[[a]](https://en.wikipedia.org/wiki/Paging#cite_note-1) for use in [main memory](https://en.wikipedia.org/wiki/Computer_data_storage#Primary_storage).[[1]](https://en.wikipedia.org/wiki/Paging#cite_note-ostep-1-2) In this scheme, the operating system retrieves data from secondary storage in same-size [blocks](https://en.wikipedia.org/wiki/Block_(data_storage)) called [*pages*](https://en.wikipedia.org/wiki/Page_(computer_memory)).
* Paging is an important part of [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) implementations in modern operating systems, using secondary storage to let programs exceed the size of available physical memory.

Demand Paging

* **demand paging** is a method of [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) management.
* In a system that uses demand paging, the operating system copies a disk [page](https://en.wikipedia.org/wiki/Paging) into physical memory only if an attempt is made to access it and that page is not already in memory (*i.e.*, if a [page fault](https://en.wikipedia.org/wiki/Page_fault) occurs).
* It follows that a [process](https://en.wikipedia.org/wiki/Process_(computing)) begins execution with none of its pages in physical memory, and many page faults will occur until most of a process's [working set](https://en.wikipedia.org/wiki/Working_set) of pages are located in physical memory.
* This is an example of a [lazy loading](https://en.wikipedia.org/wiki/Lazy_loading) technique

Types of Page Algo

* First In First Out (FIFO)
* Least Recently Used (LRU)
* Optimal Page Replacement

#### **Least Recently Used (LRU)**

Least Recently Used page replacement algorithm keeps track of page usage over a short period of time. It works on the idea that the pages that have been most heavily used in the past are most likely to be used heavily in the future too.

In LRU, whenever page replacement happens, the page which has not been used for the longest amount of time is replaced.

#### 

#### 

#### **Optimal Page Replacement**

Optimal Page Replacement algorithm is the best page replacement algorithm as it gives the least number of page faults. It is also known as OPT, clairvoyant replacement algorithm, or Belady’s optimal page replacement policy.

In this algorithm, pages are replaced which would not be used for the longest duration of time in the future, i.e., the pages in the memory which are going to be referred farthest in the future are replaced

***Advantages***

* Easy to Implement.
* Simple data structures are used.
* Highly efficient.

***Disadvantages***

* Requires future knowledge of the program.
* Time-consuming.

Enhanced Second Chance Algo

The reference bit and modify bit form a pair (r,m) where

1. (0,0) neither recently used nor modified - replace this page!

2. (0,1) not recently used but modified - not as good to replace, since the OS must write out this page, but it might not be needed anymore.

3. (1,0) recently used and unmodified - probably will be used again soon, but OS need not write it out before replacing it

4. (1,1) recently used and modified - probably will be used again soon and the OS must write it out before replacing

* On a page fault, the OS searches for the first page in the lowest nonempty class.

The OS goes around at most three times searching for the (0,0) class.

1. Page with (0,0) => replace the page.

2. Page with (0,1) => initiate an I/O to write out the page, locks the page in memory until the I/O completes, clears the modified bit, and continue the search

3. For pages with the reference bit set, the reference bit is cleared.

4. If the hand goes completely around once, there was no (0,0) page. • On the second pass, a page that was originally (0,1) or (1,0) might have been changed to (0,0) => replace this page

• If the page is being written out, waits for the I/O to complete and then remove the page. • A (0,1) page is treated as on the first pass. • By the third pass, all the pages will be at (0,0).

.