Memory Management and Virtual Memory - Logical & physical Address Space, Swapping, Contiguous Allocation, Paging, Structure of Page Table. Segmentation, Segmentation with Paging, Virtual Memory, Demand Paging, Performance of Demanding Paging, Page Replacement Page Replacement Algorithms, Allocation of Frames, Thrashing.

logical and physical addresses

An address generated by the CPU is commonly refereed as **Logical Address**,whereas the address seen by the memory unit,that is one loaded into the memory address register of the memory is commonly refereed as the **Physical Address**.The compile time and load time address binding generates the identical **logical and physical addresses**.However, the execution time address binding scheme results indiffering **logicalandphysical addresses**.

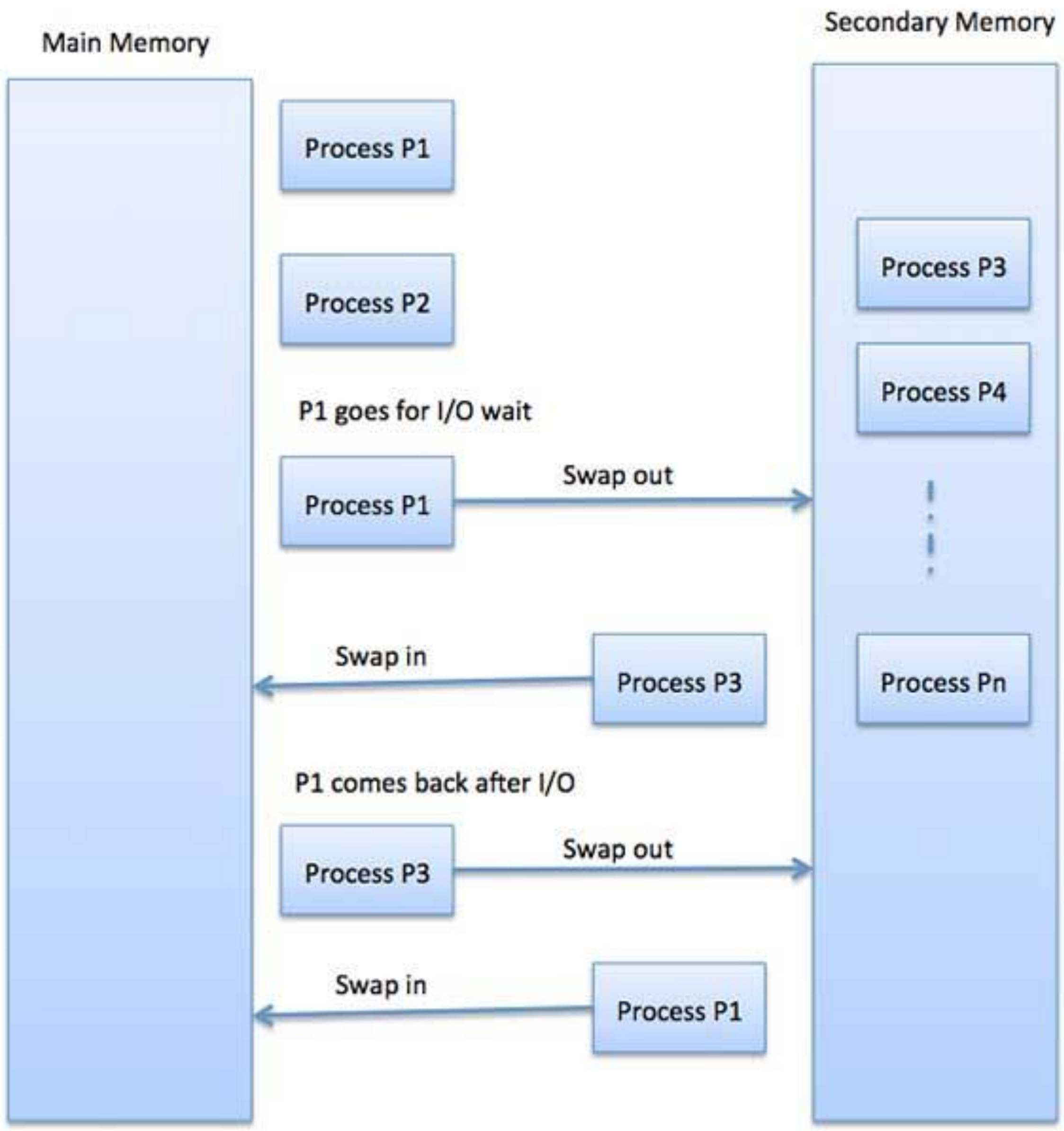
The set of all **logical addresses** generated by a program is known as **Logical Address Space**,whereas the set of all **physical addresses**corresponding to these logical addresses is **Physical Address Space**.Now, the run time mapping from virtual address to **physical address** is done by a hardware device known as **Memory Management Unit**.Here in the case of mapping the base register is known as **relocation register**.The value in the relocation register is added to the address generated by a user process at the time it is sent to memory.Let's understand this situation with the help of example:If the base register contains the value 1000,then an attempt by the user to address location 0 is dynamically relocated to location 1000,an access to location 346 is mapped to location 1346.

The user program never sees the **real physical address** space,it always deals with the **Logical addresses**.As we have two different type of addresses **Logical address** in the range (0 to max) and **Physical addresses** in the range(R to R+max) where R is the value of relocationregister.The user generates only **logical addresses** and thinks that the process runs in location to 0 to max.As it is clear from the above text that user program supplies only logical addresses,these **logical addresses**must be mapped to **physical address** before they are used.

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.

Though performance is usually affected by swapping process but it helps in running multiple and big processes in parallel and that's the reason **Swapping is also known as a technique for memory compaction**.



The total time taken by swapping process includes the time it takes to move the entire process to a secondary disk and then to copy the process back to memory, as well as the time the process takes to regain main memory.
Let us assume that the user process is of size 2048KB and on a standard hard disk where swapping will take place has a data transfer rate around 1 MB per second. The actual transfer of the 1000K process to or from memory will take

```
2048KB / 1024KB per second
= 2 seconds
= 2000 milliseconds
```

Now considering in and out time, it will take complete 4000 milliseconds plus other overhead where the process competes to regain main memory.

Contiguous memory allocation is one of the efficient ways of allocating main memory to the processes. The memory is divided into two partitions. One for the Operating System and another for the user processes. Operating System is placed in low or high memory depending on the interrupt vector placed. In contiguous memory allocation each process is contained in a single contiguous section of memory.

Memory allocation

There are two methods namely, multiple partition method and a general fixed partition method. In multiple partition method, the memory is divided into several fixed size partitions. One process occupies each partition. This scheme is rarely used nowadays. Degree of multiprogramming depends on the number of partitions. Degree of multiprogramming is the number of programs that are in the main memory. The CPU is never left idle in multiprogramming. This was used by IBM OS/360 called MFT. MFT stands for Multiprogramming with a Fixed number of Tasks.

Generalization of fixed partition scheme is used in MVT. MVT stands for Multiprogramming with a Variable number of Tasks. The Operating System keeps track of which parts of memory are available and which is occupied. This is done with the help of a table that is maintained by the Operating System. Initially the whole of the available memory is treated as one large block of memory called a **hole**. The programs that enter a system are maintained in an input queue. From the hole, blocks of main memory are allocated to the programs in the input queue. If the hole is large, then it is split into two, and one half is allocated to the arriving process and the other half is returned. As and when memory is allocated, a set of holes in scattered. If holes are adjacent, they can be merged.

Now there comes a general dynamic storage allocation problem. The following are the solutions to the dynamic storage allocation problem.

- First fit: The first hole that is large enough is allocated. Searching for the holes starts from the beginning of the set of holes or from where the previous first fit search ended.

- Best fit: The smallest hole that is big enough to accommodate the incoming process is allocated. If the available holes are ordered, then the searching can be reduced.

- Worst fit: The largest of the available holes is allocated.

First and Best fits decrease time and storage utilization. First fit is generally faster.
Fragmentation

The disadvantage of contiguous memory allocation is **fragmentation**. There are two types of fragmentation, namely, Internal fragmentation and External fragmentation.

Internal fragmentation

When memory is free internally, that is inside a process but it cannot be used, we call that fragment as internal fragment. For example say a hole of size 18464 bytes is available. Let the size of the process be 18462. If the hole is allocated to this process, then two bytes are left which is not used. These two bytes which cannot be used forms the internal fragmentation. The worst part of it is that the overhead to maintain these two bytes is more than two bytes.

External fragmentation

All the three dynamic storage allocation methods discussed above suffer external fragmentation. When the total memory space that is got by adding the scattered holes is sufficient to satisfy a request but it is not available contiguously, then this type of fragmentation is called external fragmentation.

The solution to this kind of external fragmentation is compaction. **Compaction** is a method by which all free memory that are scattered are placed together in one large memory block. It is to be noted that compaction cannot be done if relocation is done at compile time or assembly time. It is possible only if dynamic relocation is done, that is relocation at execution time.

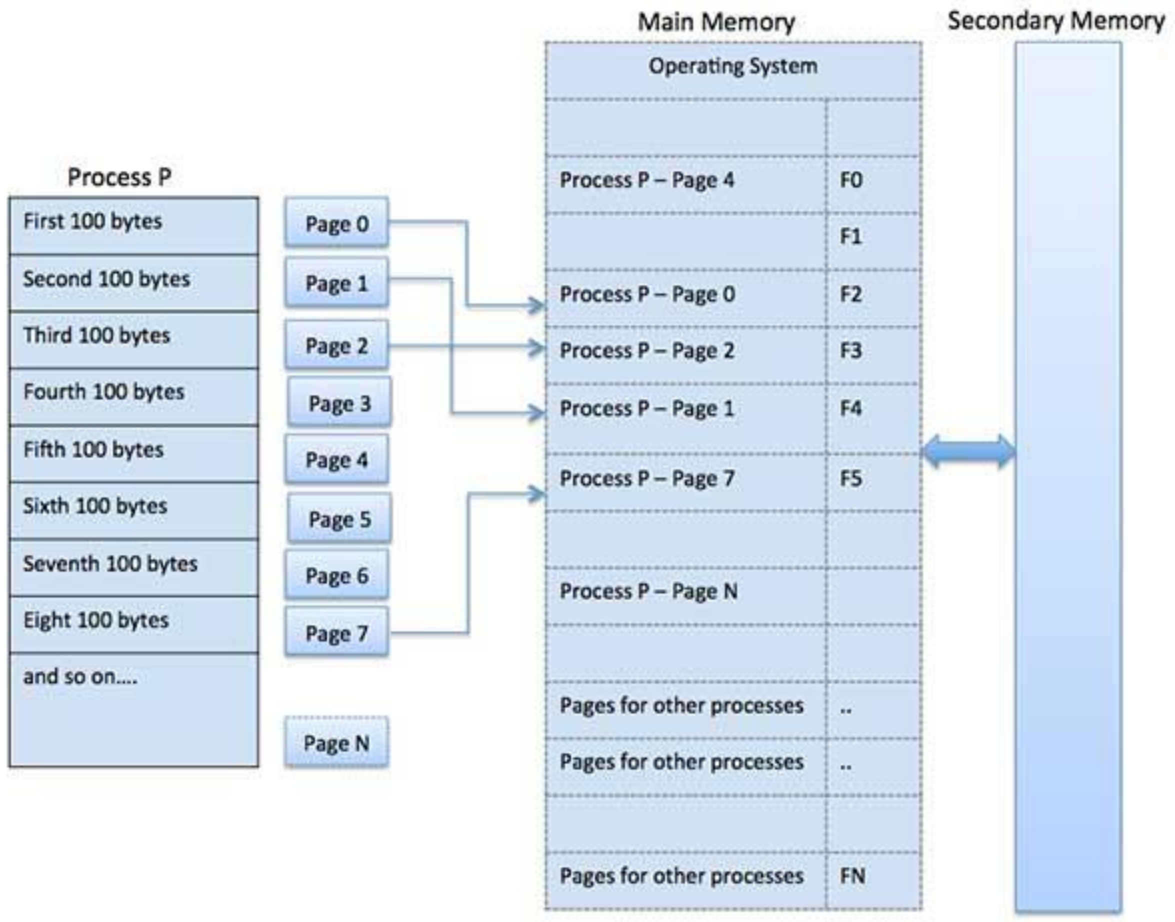
One more solution to external fragmentation is to have the logical address space and physical address space to be non contiguous. Paging and Segmentation are popular non contiguous allocation methods.

Paging

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 that's set up to emulate the computer's RAM. Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process address space is broken into blocks of the same size called **pages** (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation.



Here is a list of advantages and disadvantages of paging –

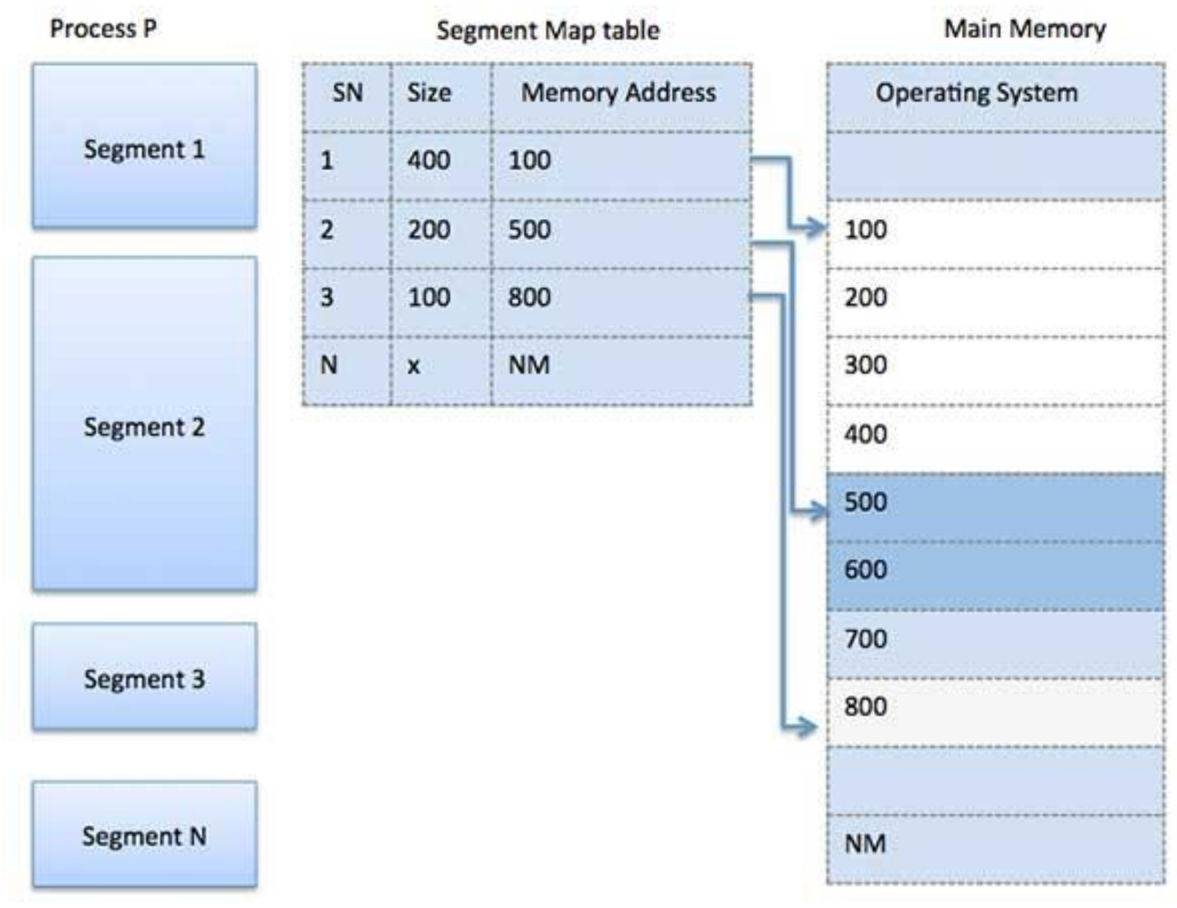
- Paging reduces external fragmentation, but still suffer from internal fragmentation.

- Paging is simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.
- Page table requires extra memory space, so may not be good for a system having small RAM.

Segmentation

Segmentation is a memory management technique in which each job is divided into several segments of different sizes, one for each module that contains pieces that perform related functions. Each segment is actually a different logical address space of the program.

Segmentation memory management works very similar to paging but here segments are of variable-length where as in paging pages are of fixed size.
A program segment contains the program's main function, utility functions, data structures, and so on. The operating system maintains a **segment map table**for every process and a list of free memory blocks along with segment numbers, their size and corresponding memory locations in main memory. For each segment, the table stores the starting address of the segment and the length of the segment. A reference to a memory location includes a value that identifies a segment and an offset.



Virtual Memory

Virtual Memory is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory. This technique is useful as large virtual memory is provided for user programs when a very small physical memory is there.

In real scenarios, most processes never need all their pages at once, for following reasons :

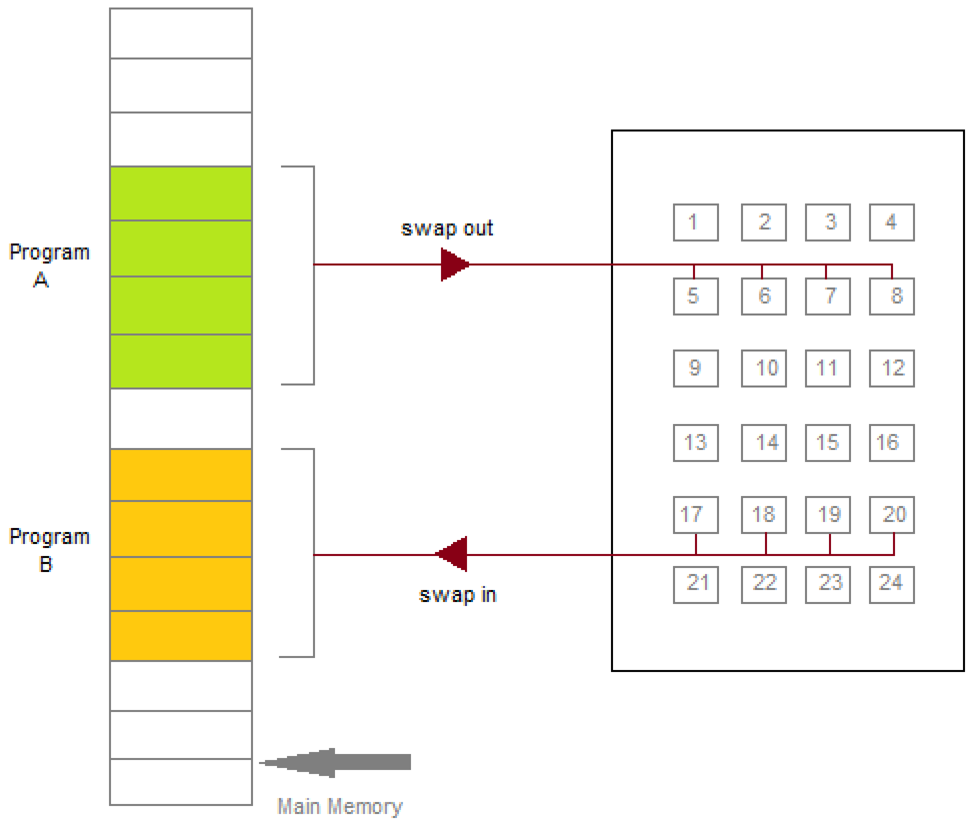
- Error handling code is not needed unless that specific error occurs, some of which are quite rare.
- Arrays are often over-sized for worst-case scenarios, and only a small fraction of the arrays are actually used in practice.
- Certain features of certain programs are rarely used.

Benefits of having Virtual Memory :

1. Large programs can be written, as virtual space available is huge compared to physical memory.
2. Less I/O required, leads to faster and easy swapping of processes.
3. More physical memory available, as programs are stored on virtual memory, so they occupy very less space on actual physical memory.

Demand Paging

The basic idea behind demand paging is that when a process is swapped in, its pages are not swapped in all at once. Rather they are swapped in only when the process needs them(On demand). This is termed as lazy swapper, although a pager is a more accurate term.



Initially only those pages are loaded which will be required the process immediately.

The pages that are not moved into the memory, are marked as invalid in the page table. For an invalid entry the rest of the table is empty. In case of pages that are loaded in the memory, they are marked as valid along with the information about where to find the swapped out page.
When the process requires any of the page that is not loaded into the memory, a page fault trap is triggered and following steps are followed,

1. The memory address which is requested by the process is first checked, to verify the request made by the process.
2. If its found to be invalid, the process is terminated.
3. In case the request by the process is valid, a free frame is located, possibly from a free-frame list, where the required page will be moved.
4. A new operation is scheduled to move the necessary page from disk to the specified memory location. (This will usually block the process on an I/O wait, allowing some other process to use the CPU in the meantime.)
5. When the I/O operation is complete, the process's page table is updated with the new frame number, and the invalid bit is changed to valid.
6. The instruction that caused the page fault must now be restarted from the beginning.

There are cases when no pages are loaded into the memory initially, pages are only loaded when demanded by the process by generating page faults. This is called **Pure Demand Paging**.
The only major issue with Demand Paging is, after a new page is loaded, the process starts execution from the beginning. Its is not a big issue for small programs, but for larger programs it affects performance drastically.

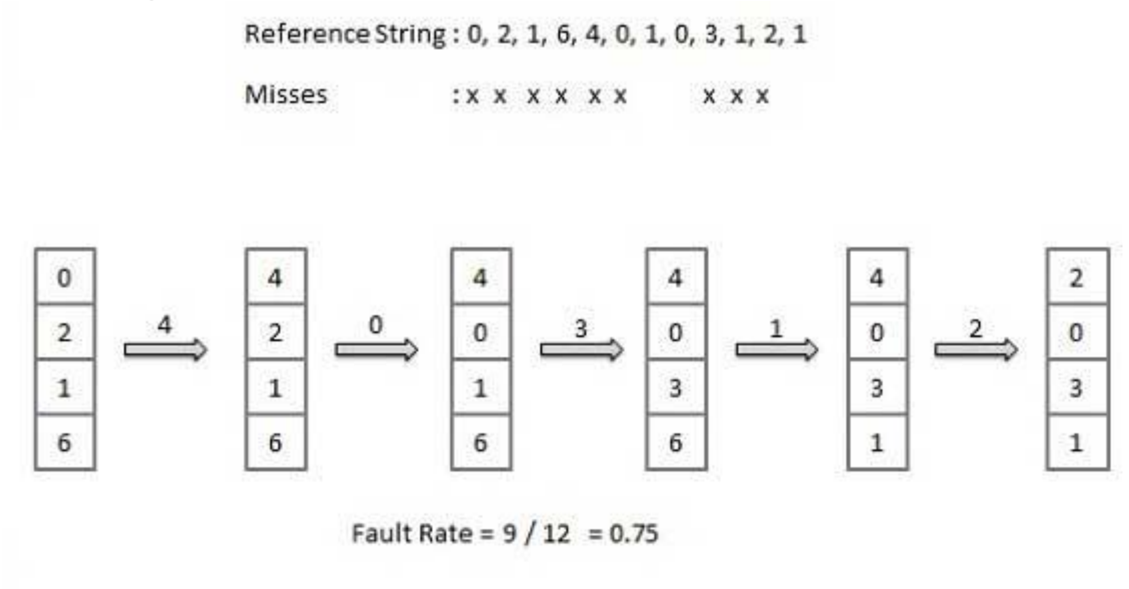
<u>Page Replacement</u>

Page Replacement Algorithms

- 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

1.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.



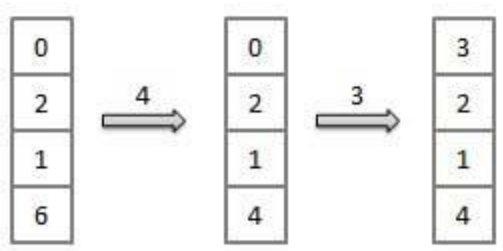
Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1
Misses : x x x x x x x x x
0 2 1 6
4
4 2 1 6
0
4 0 1 6
3
4 0 3 6
1
4 0 3 1
2
2 0 3 1
Fault Rate = 9 / 12 = 0.75

2.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



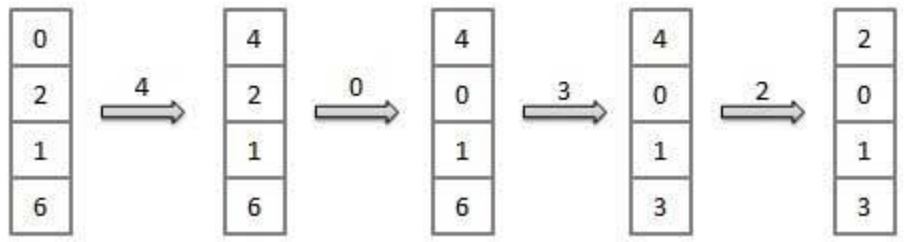
Fault Rate = 6 / 12 = 0.50

3.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

4.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.

Allocation of Frames in Operating System:

Normally, there are fixed amounts of free memory with various processes at different time in a system. The question is how this fixed amount of free memory is allocated among the different processes.

The simplest case is the single process system. All available memory for user programs can initially be put on the free frame list (pure demand paging). When the user program starts its execution, it will generate a sequence of page faults. The user program would get all free frames from the free frame list. As soon as this list was exhausted, and the more free frames are required, the page replacement algorithm can be used to select one of the in-used pages to be replaced with the next required page and so on. After the program was terminated, all used pages are put on the free frame list again.

The frame allocation procedure is more complicated when there are two of more programs in memory at the same time.