Memory Management in Linux

What are the 3 areas of memory management?
Memory management operates at three levels: hardware, operating system and program/application.
The management capabilities at each level work together to optimize memory availability and efficiency.

Degree of Multiprogramming

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 is called Memory Management.

The degree of multiprogramming describes the maximum number of processes that a single-processor system can accommodate efficiently.

bring more and more process from secondary memory to primary(RAM) so that utilization of CPU increased.and hence efficency of system gets increased.

Higher the degree of multiprogramming higher the utilization of CPU.

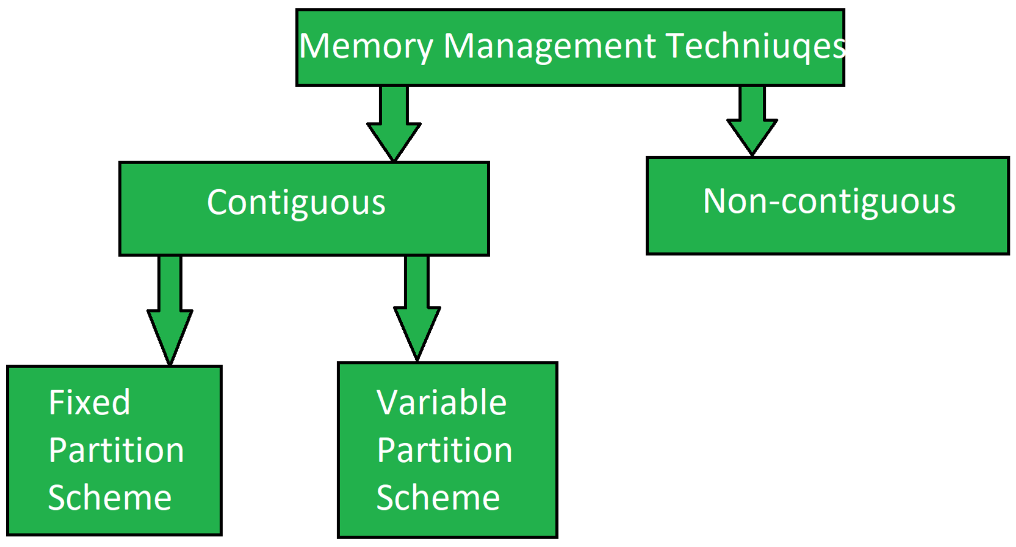
process executed in two ways either on CPU or want to make some i/o request

lets take an example

1.
let say Ram of size 4mb and a process of size 4mb
then ony 1 process fits in ram
and say K is the time for which it performes i/o operation
it executed on CPU= 1-K time
therefore CPU Utilization= 1-K lets say 70% it perform i/o operation
Then cpu utilization= 30%

2.
let say Ram of size 8mb and a process of size 4mb
2 process can fits on RAM
CPU utilization= 1-K*k =(1-.49)=51%

3.
let say Ram of size 16mb and a process of size 4mb
4 process can fits on RAM
CPU Utilization = 1-K*K*K*K =(1-.2401)=76%



Methods to implement

1.Fixed Partitioning (Static Partition)
used in 1960's
here we can allocate contiguous memory allocate to RAM with limited and
fixed size slots or different size slots.

problems with this techinique
1.Internal Fragementation
2.Limit in Process size
3.Limitition on Degree of Multiprogramming
4.External Fragementation

External Fragmentation-
  Although we are having availability of memory in different slots and combination
but we cannot allocate new process in that available memory.

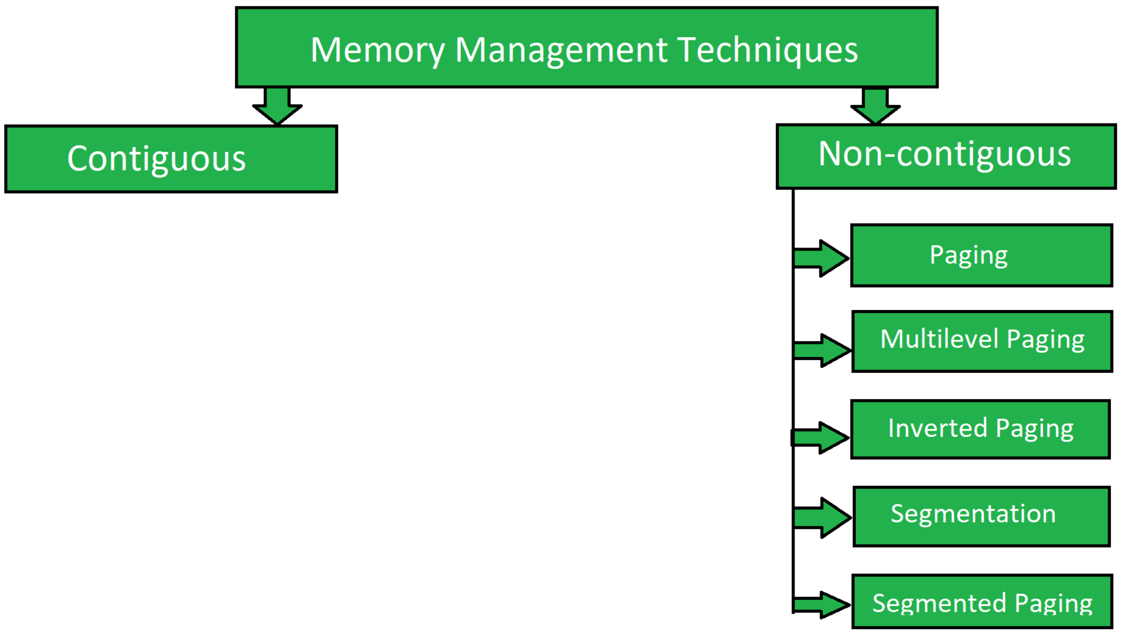
advanteges of this method is simple and easy to implement
allocation and deallocation of memory is easy.

2.Variable size Partioning:-
Variable size partitioning is part of the contiguous allocation technique. It is used to alleviate the problem faced by fixed partitioning. As opposed to fixed partitioning, in variable partitioning, partitions are not created until a process executes.

Various allocation methods in Contiguous Memory Management

First Fit: Allocate the first hole that is big enough.
Next Fit: Same as first fit but start search always from last allocated hole.
Best Fit: Allocate the smallest hole that is big enough.
Worst fit: Allocate the largest hole.

.



Paging

Paging is a memory management scheme that eliminates the need for a 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 a 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 bits,
then Logical Address Space = 2^31 Bytes = 2 GBytes (1 G = 2^30)

If Logical Address Space = 128 MByte = 2^7 * 2^20 Bytes,
then Logical Address = log$_2$ 2^27 = 27 bits

If Physical Address = 22 bits,
then Physical Address Space = 2^22 Bytes = 4 MBytes (1 M = 2^20)

If Physical Address Space = 16 MBytes = 24 * 2^20 Bytes,
then Physical Address = log 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 the paging technique.

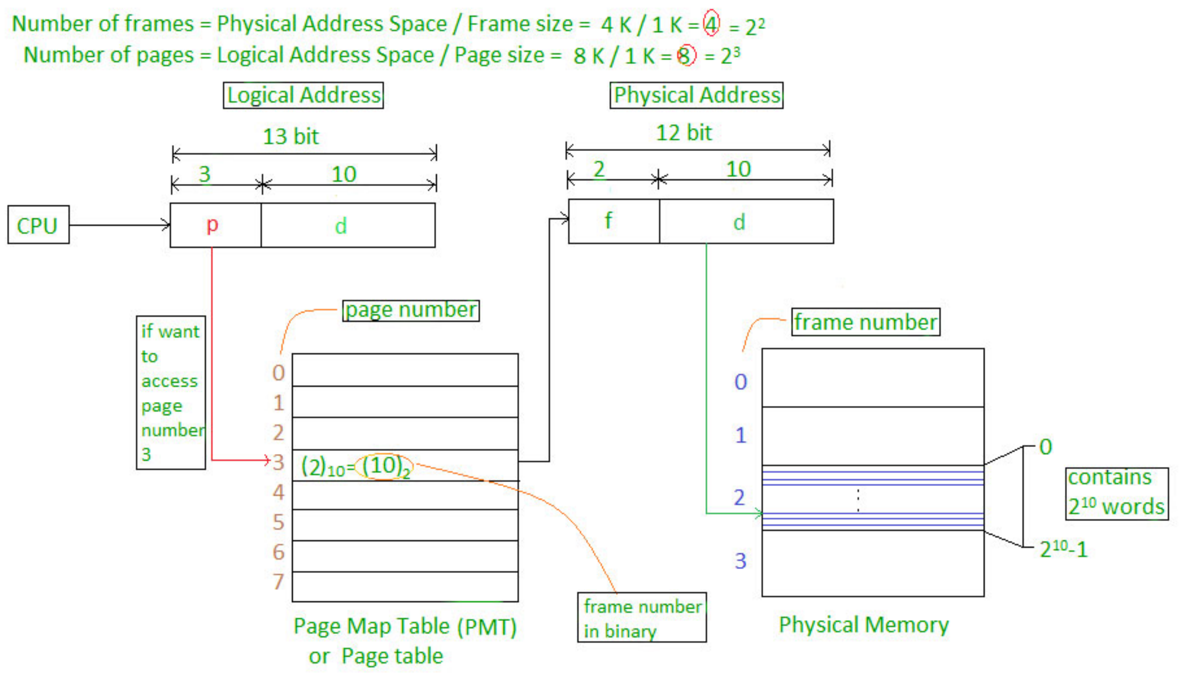
The Physical Address Space is conceptually divided into several fixed-size blocks, called frames.

The Logical Address Space is also split 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)


Number of frames = Physical Address Space / Frame size = 4 K / 1 K = 4 = 2^2
Number of pages = Logical Address Space / Page size = 8 K / 1 K = 8 = 2^3
Logical Address
13 bit
3
10
CPU
p
d
Physical Address
12 bit
2
10
f
d
if want to access page number 3
page number
0 1 2 3 4 5 6 7
$(2)_{10}$=$(10)_2$
Page Map Table (PMT) or Page table
frame number in binary
frame number
0 1 2 3
0
contains 2^{10} words
2^{10}-1
Physical Memory

The address generated by the 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 a 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

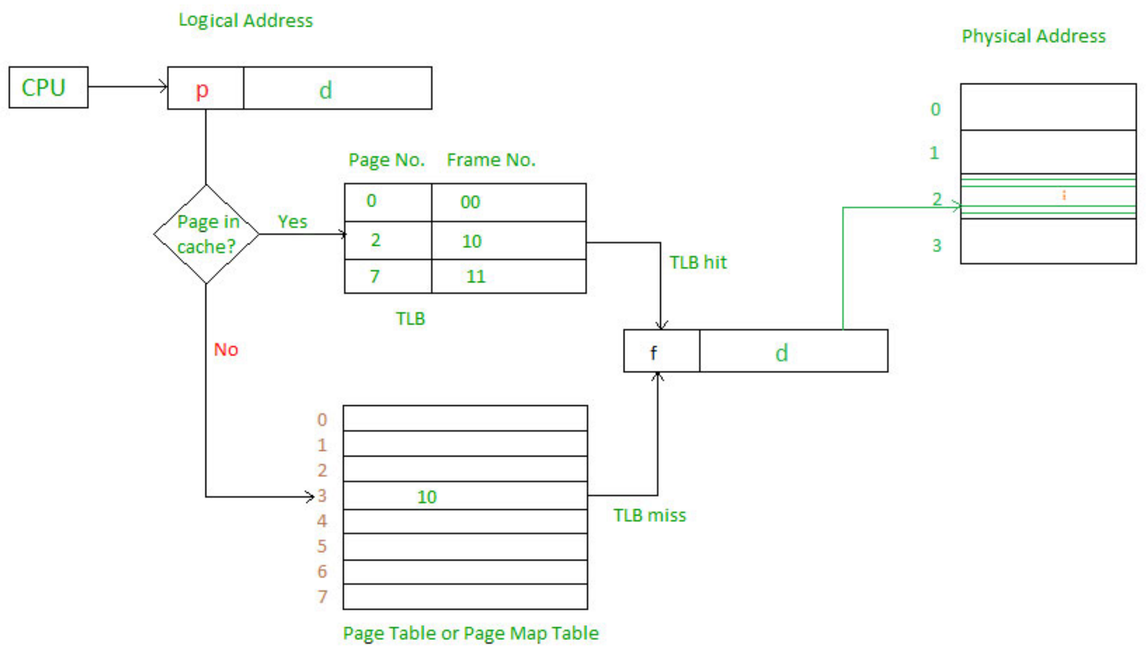
Frame Offset(d): Number of bits required to represent a particular word in a frame or frame size of Physical Address Space or word number of a frame or frame offset.

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

The TLB is an associative, high-speed memory.

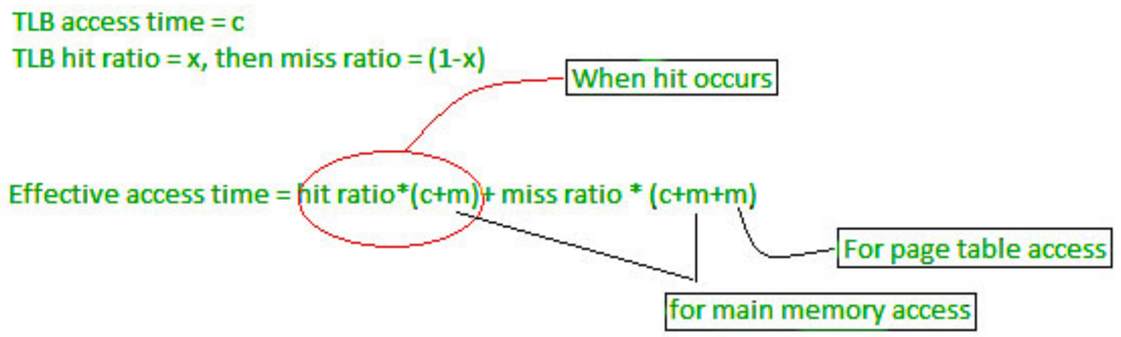
Each entry in TLB consists of two parts: a tag and a value.

When this memory is used, then an item is compared with all tags simultaneously. If the item is found, then the corresponding value is returned.



Main memory access time = m if page table are kept in main memory,

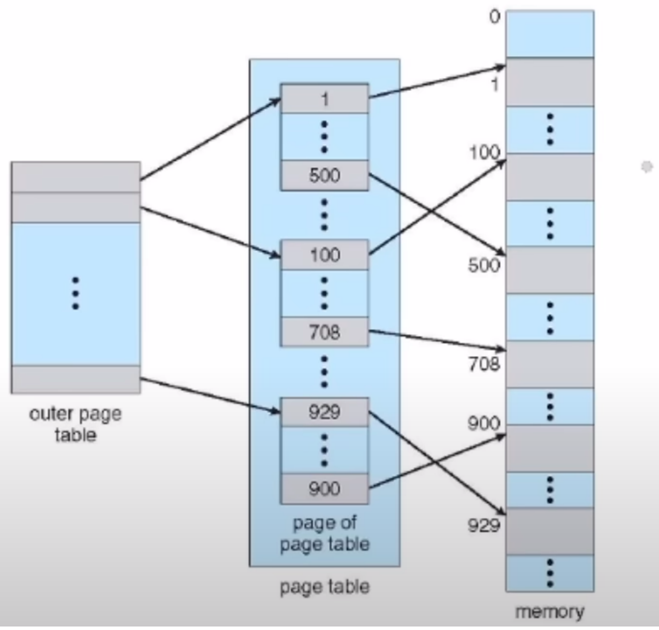
Effective access time = m(for page table) + m(for particular page in page table)


TLB access time = c

TLB hit ratio = x, then miss ratio = (1-x)

Effective access time = hit ratio*(c+m)+ miss ratio * (c+m+m)

Two-Level Page-Table Scheme



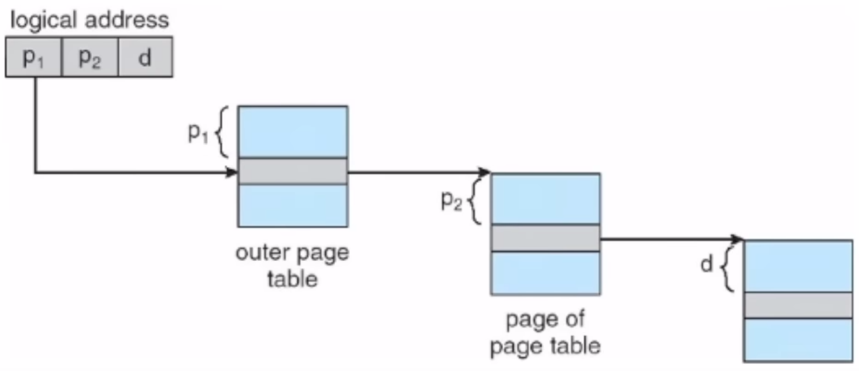
Two-Level Paging Example

- A logical address (on 32-bit machine with 1K page size) is divided into:
  - a page number consisting of 22 bits
  - a page offset consisting of 10 bits
- Since the page table is paged, the page number is further divided into:
  - a 12-bit page number
  - a 10-bit page offset
- Thus, a logical address is as follows:

page number		page offset
p_1	p_2	d
12	10	10

- where p_1 is an index into the outer page table, and p_2 is the displacement within the page of the inner page table

Address-Translation Scheme


logical address
p₁
p₂
d
p₁
outer page table
p₂
page of page table
d