5) File System and Memory Management

Page No. 66
Date

Basic Memory Management :-

Memory management is all about managing the memory of computer efficiently and effectively. If memory management is done properly, CPU and I/O utilization of system increases.

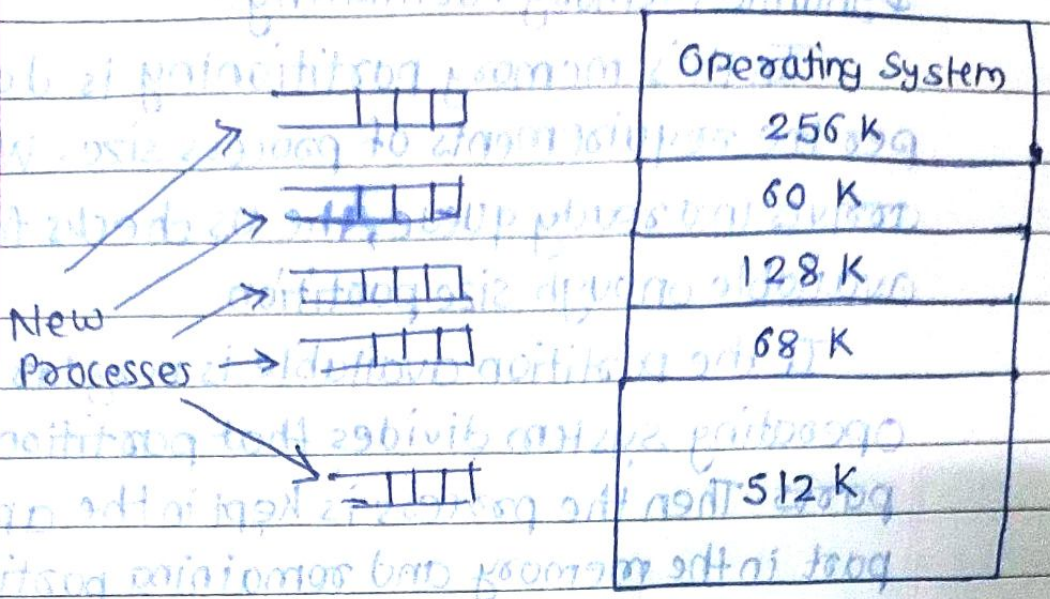
Memory Partioning :- It is one of the memory management techniques. This method divides the memory into several small parts. These parts are called as partitions.

Types of memory partitioning:
1. Static Memory Partitioning (Fixed Partitioning)
2. Dynamic Memory Partitioning (Variable Partitioning)

Static Memory Partitioning :-

Simplest method for allocating memory is to divide memory into several fixed size partitions. As the name suggests, partitions are of fixed size.

As shown in figure on next page, there are different size fixed partitions. Partition 1 is of size 60K, partition 2 is of size 128K, etc.



As partition gets free the next job or process is selected. Static or fixed partion descripter table will contain following information.

Base address	Offset

Advantages:

① Partition size is fixed and can be easily accessible for the processes thase matches with the partition size.

② No complicated algorithms required.

Disadvantages:-

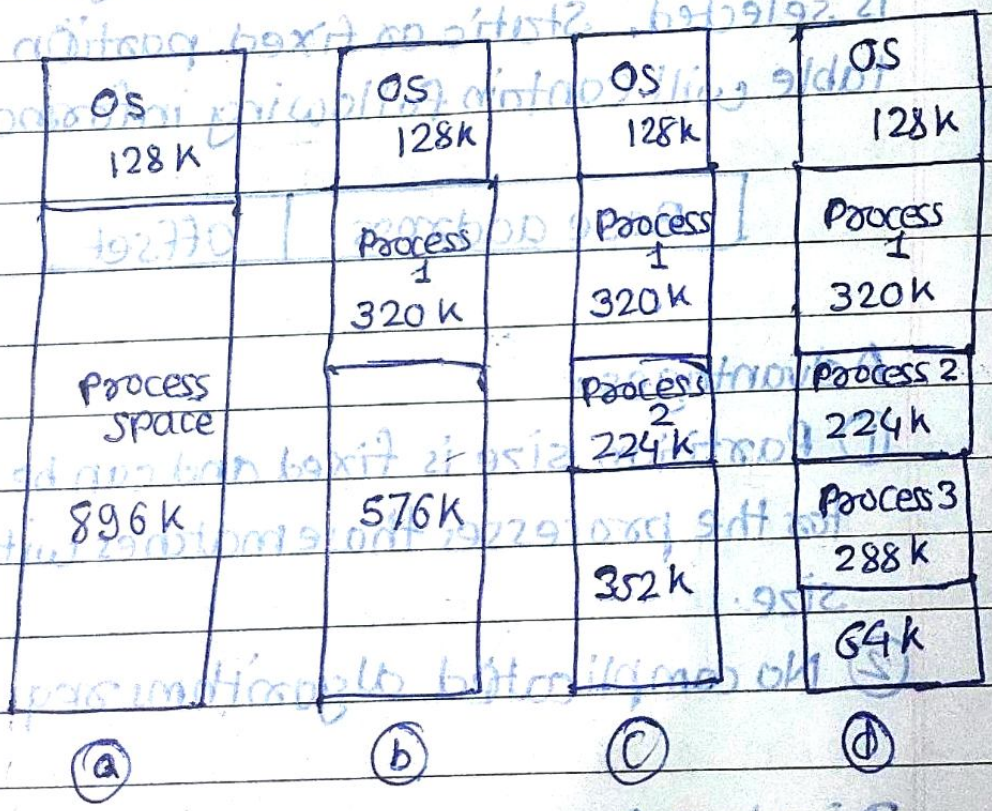
① Memory wastage

② If memory required for a process is less than size of partition, remaining part of memory cannot be allocate to any other process

Dynamic Memory Partitioning:-

In this memory partitioning is done as per the requirements of process size. When process arrives in a ready queue, the OS checks for the available enough size partition.

If the partition available is very large then operating system divides that partition into parts. Then the process is kept in the appropriate part in the memory and remaining partition of memory is included in the partition availability list.



Advantages:

1. Partitions are made as per process length
2. Memory wastage is very less
3. Maximum part of memory is utilized.

Disadvantages:

① External fragmentation occurs. To overcome this problem, we can use compaction

② In compaction, all free spaces join together to form a new block or partition for re-utilization.

Free Space Management Techniques :-

Bit map / Bit Vector :-

Generally the free space list is implemented as a bit map or bit vector. Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.

Ex :- Consider a disk where 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26 and 27 are free and rest are allocated.

Then the free space bitmap would be

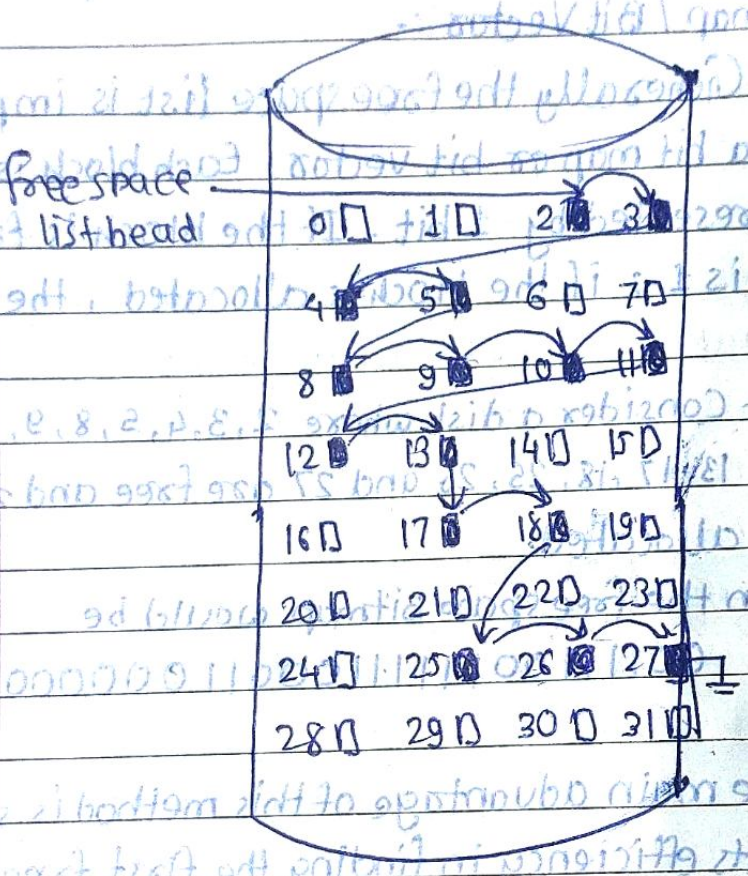
001111001111000110000001100000...

The main advantage of this method is simplicity and its efficiency in finding the first free block of n consecutive free blocks on the disk.

Disadvantage :- Unfortunately bit vectors are inefficient unless the entire vector is kept in main memory. Keeping it in main memory is possible for smaller disks only.

Linked list :- In this, we link together all the free disk blocks, keeping the pointer to the next free disk block and so on.

Example :- Blocks 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 17, 18, 25, 26, 27 are free and the rest of blocks are allocated.



Disadvantages :- This scheme is not efficient; to traverse the list, we must read each block, which requires more I/O time.

Fortunately, traversing the free list is not required frequently. Operating System simply needs a free blocks so that it can allocate that block to the file, so the first block in the free list is used.

Date

File :- A file is a named collection of related information stored on secondary storage. A file is a sequence of bits, bytes, lines or records. Files represent programs and data.

A file has a certain defined structure, which depends on its type:

- Text file is a sequence of characters organised into lines.
- Source file is a sequence of functions, each of which is further organised as declarations followed by executable statements.
- Executable file is a series of code sections that the loader can bring into memory and execute.

Attributes of file:

1. Name :- The symbolic file name is the only information kept in human readable form.
2. Identifier :- This unique tag, usually a number, identifies the file within the file system. It is the non-human-readable name for the file.
3. Type :- This information is needed for systems that support different types of file.
4. Location :- This information is a pointer to the device and to the location of that file to the device.
5. size :- The current size of files and possibly the maximum allowed size are included in this attribute.

Date

⑥ Protection :- Access control information determines who can do reading, writing or executing, and so on.

⑦ Time date and user identification :- This information may be kept for creation, last modification and last use. This data can be useful for protection security and usage monitoring.

⑧ Operations on file :-

Six basic file operations :-

① Writing a file :- To write a file, we make a system call specifying both the name of the file and information to be written to the file. Given the name of the file, the system searches the directory to find the file's location. The system must keep a write pointer to the location in the file where the next write is to take place. It must be updated whenever write occurs.

② Creating a file :- Two steps are necessary to create a file :

- First, space in the file system must be found for file
- Second, an entry for the new file must be made in the directory.

③ Reading a file :- To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of file should be put. Again the directory is searched for associated entry, and the system needs to keep a read pointer to the location in the file where the next read is to take place. Once the read has taken place, the read pointer is updated.

④ Repositioning within a file :- The directory is searched for appropriate entry and the current-file-position-pointer is repositioned to a given value. This file operation is also known as seek.

⑤ Deleting a file :- To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files.

⑥ Truncating a file :- The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then create it. This function allows all attributes to remain unchanged - except for file length - but lets the file be reset to length zero and its file space released.

File Types :-

When we design a file, we need to consider whether the operating system recognizes and support that file types. If an OS recognises the type of a file, it can then operate on the file in proper ways. A common technique for implementing file types is to include a type as part of a file name.

The name is split into two parts:

① Name

② Extension seperated by a period (.)

The System uses extension to indicate :

① Type of the file

② The type of operations that can be done on that file.

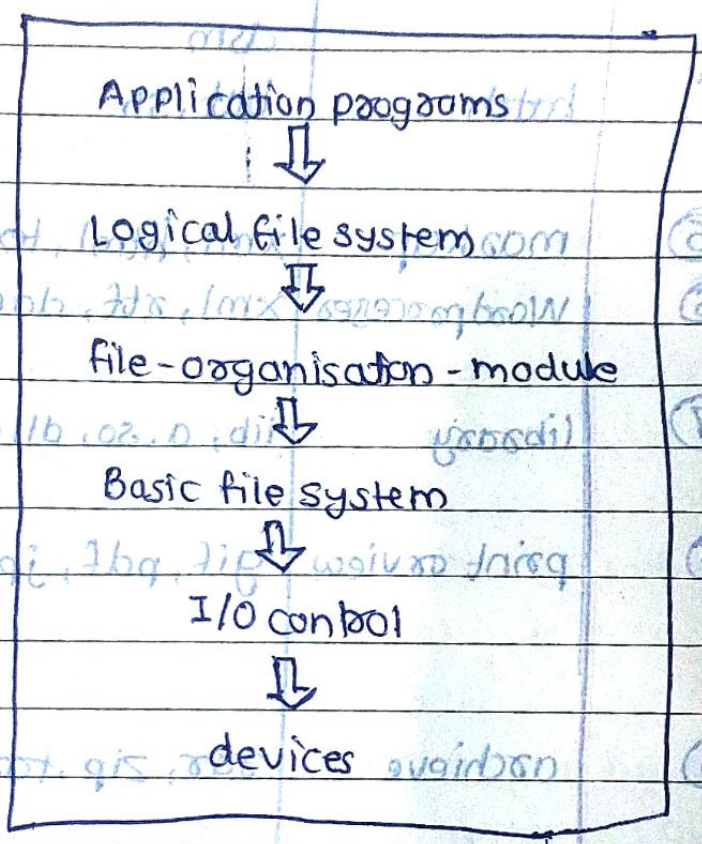
Only a file with .com, .exe or .sh can be executed. The .com and .exe are two forms of binary executable files, whereas .sh file is a shell script containing, in ASCII format, command to the operating system

	file type	usual extension	function
①	executable	exe, com, bin or none	ready to run machine language programs
②	object	obj, o	compiled, machine language, not linked
③	source code	c, cc, java, perl, asm	source code in various languages
④	batch	bat, sh	commands to command interpreter
⑤	markup	xml, html, tex	textual data, documents
⑥	word processor	xml, rtf, docx	various word processor formats
⑦	library	lib, a, so, dll	libraries of routines for programmers
⑧	print or view	gif, pdf, jpg	ASCII binary file in a format for printing or viewing.
⑨	archieve	rar, zip, tar	related files grouped into one file, sometime compressed for archiving or storage
⑩	multimedia	mpeg, mov, mp3 mp4, avi	binary file containing audio or A/V information

Date

File System Structures :-

File system provides efficient and convenient access to the disk by allowing data to be stored, located and retrieved easily. File system is generally composed of many levels. The structure is shown in figure below :



Each level in the design uses the feature of lower levels to create new features for use by higher levels

I/O control level :- It consists of device drivers and interrupts handlers to transfer information between the main memory and the disk system. A device driver translate higher level commands into low level instructions that are used by the hardware controller.

Basic file system :-

It intends to :

① Issue generic commands to the appropriate device driver to read and write physical blocks on the disk.

② Manages the memory, buffers and caches that hold various file-systems, directory and data blocks

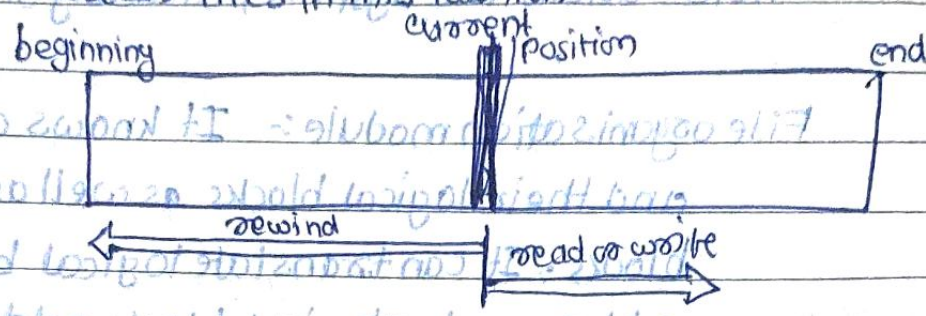
File organisation module :- It knows about files and their logical blocks as well as physical blocks. It can translate logical block addresses to physical block addresses.

Logical file system :- It manages metadata information. Metadata includes all of the file system structure except the actual data. It manages the directory structure. It maintains file structure via file control blocks

When a layered structured is used for file-system implementation, duplication of code is minimized. But layering can introduce more OS overhead, which may result in decreased performance.

Access methods :-

(1) Sequential access :- The simplest and most common access method is sequential access. Information in the file is processed in order, one record after the other. Editors and compilers usually access files in this fashion.



- read next() - reads the next portion of the file and automatically advances a file pointer, which tracks the I/O location
- write next() - appends to the end of the file and advances to the end of newly written material

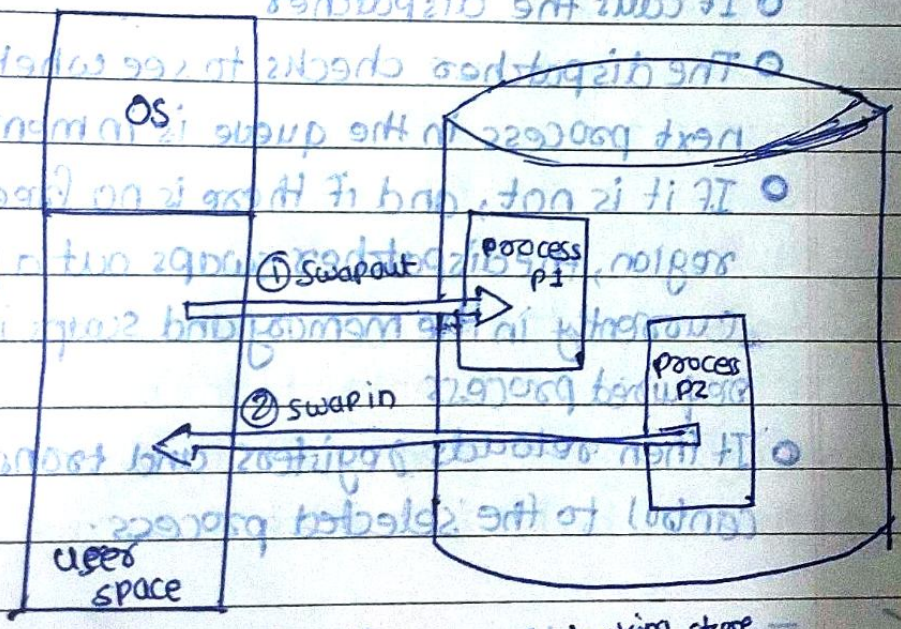
Such a file can be reset to the beginning, and on some systems, a program may be able to to skip forward or backward. Sequential access is based on tape model of the file and works well on sequential access devices as it does on random access ones.

(2) Direct/Relative/Random access :- The direct access method is based on a disk model of a file. For direct access, the file is viewed as a numbered sequence blocks or records. Direct access file are of great use for fast access to large amount of information. This approach is mainly used by databases.

for direct access method, the file operations must be modified to include a block number as a parameter. Thus, we have read(n), where n is the block number, rather than read next() and write(n) rather than write next().

Sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp;
	cp = cp + 1;
write_next	write cp;
	cp = cp + 1;

Swapping :-



A process can be swapped temporarily out of the memory to a backing store and then brought back into memory for continued execution

Swapping makes it possible for the total physical address space of all processes to exceed the real physical memory of the system; it increases the degree of multiprogramming.

Standard swapping involves moving processes between main memory and the backing store. The backing store is generally hard disk and it must be large enough.

The system maintains a ready queue consisting of all processes whose memory images are on the backing store or in memory ready to run.

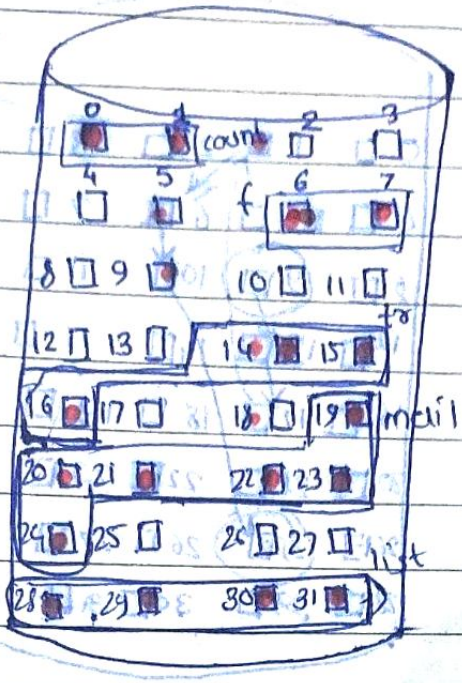
Whenever CPU scheduler deceides to execute a process:

- It calls the dispatcher
- The dispatcher checks to see whether the next process in the queue is in memory.
- If it is not, and if there is no free memory region, the dispatcher swaps out a process currently in the memory and swaps in the required process.
- It then reloads registers and transfers control to the selected process.

File allocation methods:

1. Contiguous allocation
2. Linked allocation
3. Indexed allocation

① Contiguous Allocation



directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Contiguous allocation requires that each file occupy set of contiguous blocks on the disk. The directory entry for each file indicates the address of the starting block and length of the area allocated for this file.

Advantages:

① Disk addresses define a linear ordering on the disk.

② Accessing a file that is contiguously allocated is easy

③ It can support both sequential & direct access.

Disadvantages:

① Difficulty in finding space for a file.

② Dynamic storage allocation problem

③ It may suffer a problem of external fragmentation

④ Determining how much space needed for file is difficult

⑤ Compaction may be required.

② Linked allocation :-

directory		
file	start	end
jeep	9	25

With linked allocation, each file is a linked list of disk blocks; the disk blocks may be scattered anywhere on the disk.

Here, each block contains a pointer to the next block. Thus if each block is 512 bytes in size, and a disk address (the pointer) requires 4 bytes, then the user sees blocks of 508 bytes.

A write to a file needs to find a free block is written to and is linked to the end of the file. To read a file, we simply read blocks by following the pointers from block to block.

Advantages :

① Linked allocation solves many problems of contiguous allocation.

② No external fragmentation

③ The size of a file need not to be declared when the file is created.

④ A file can continue to grow as long as free blocks are available.

⑤ Compaction is not required

Disadvantages:-

① It is unreliable because if pointer is lost, a problem can occur.

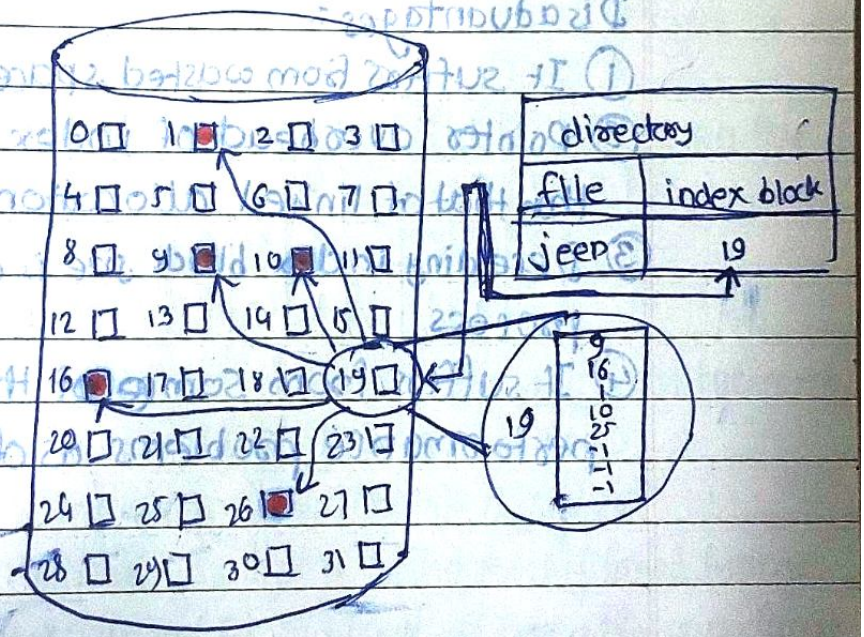
② It can be used effectively only for sequential access files

③ Each access to a pointer requires a disk read, some require disk seek.

④ It is inefficient to support a direct access-capability

⑤ The space required for pointers is considerable.

③ Indexed Allocation :-



Indexed allocation solves the problem of ~~inefficient~~ inefficient direct access by bringing all the pointers together into one location called index block. Each file has its own index block, which is an array of disk block addresses.

Date

To find and read the n^{th} block, we use the pointer in the n^{th} index block entry. When the file is created, all pointers in this index are set to null. When n^{th} block is first written, a block is obtained from the free space manager and its address is put in the n^{th} index block entry.

Advantages :-

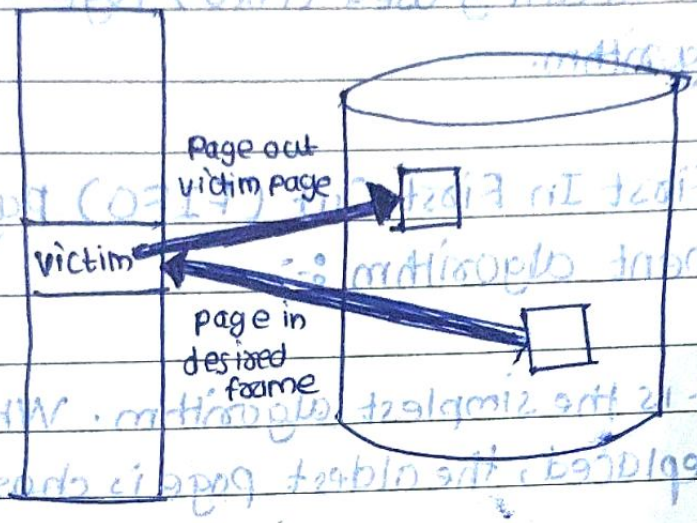
1. Faster than contiguous and linked allocation
2. It supports direct access
3. No external fragmentation
4. Accessing index block is easy

Disadvantages :-

1. It suffers from wasted space
2. Pointer overhead of index block is greater than that of linked allocation.
3. Deciding index block size is a complicated process
4. It suffers from some of the same performance problems as of linked allocation

Page replacement

Main memory is divided into number of frames. If no frame is free, we find one that is not currently being used and free it. We can now use this freed frame to hold the required new page. This technique is called as page replacement



Page replacement process :-

① Find the location of a desired new page on the disk.

② Find a free frame :

a] If there is a free frame, use it.

b] If there is no free frame, use a page replacement algorithm to select a victim frame

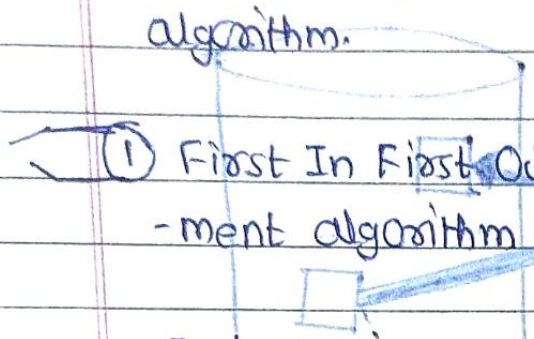
c] Write the victim frame to the disk.

③ Read the desired page into the newly freed frame

④ Continue the user process from where the page fault occured

Page replacement algorithms:

1. First in First out (FIFO) page replacement algorithm
2. Optimal (OPT) Page replacement algorithm
3. Least Recently Used (LRU) Page replacement algorithm
4. Not Recently Used (NRU) Page replacement algorithm.



1. First In First Out (FIFO) Page replacement algorithm :-

It is the simplest algorithm. When page is to be replaced, the oldest page is chosen.

Example

String	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
1	7	7	7	2		2	2	4	4	4	0			1	0			7	7	7
2		0	0	0		3	3	3	2	2	2			1	1			1	0	0
3			1	1		1	0	0	0	3	3			3	2			2	2	1

Our three frame initially are empty. The first three references will cause page fault (7, 0, 1) and they will be brought into empty frames.
The next reference (2) replaces 7, because it was the oldest one among the three pages in memory. Since next reference 0 is already in this memory, no fault for this reference.
The process continues.
Page faults = 15 Page hits = 5

Advantages :-

① Simplest and easy to implement
② Works in sequential manner

Disadvantages :-

① Slow
② Page fault rate is high
③ FIFO may replace pages that are just about to be used again.

② Optimal (OPT) Page Replacement Algorithm :

It replaces the page that will not be used for the longest period of time

Example :

String	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
①	7	7	7	2		2		2			2			2				7		
②		0	0	0		0		4			0			0				0		
③			1	1		3		3			3			1				1		

Q. In three frames are initially empty. The first three references (7, 0, 1) will cause page faults and are brought into empty frame. Reference 2 replaces 7 because, 7 will not be used for a longest time. Also Reference to 3 will replace 1 as 1 will not be used for longest time. This process continues till end.

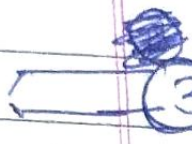
Page faults = 9. Page hits = 11

Advantages:-

① Lowest page fault rate
② Fast and efficient

Disadvantages:-

① Difficult to implement
② Advanced (future) knowledge of reference string is required.



③ Least Recently Used (LRU) Page replacement algorithm:-

LRU algorithm uses the the time when page has not been used in the past. It replaces the page that has not been used for longest period of time.

Example:

String:-

	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
①	7	7	7	2		2		4	4	4	0			1			1			
②		0	0	0		0		0	0	3	3			3		0		0		
③			1	1		3		3	2	2	2			2		2		7		

Our three frames are initially empty. The first three references (7, 0, 1) cause page faults and are brought into empty frames. Here, first five frames are as same as those for optimal replacement. Next reference 4 replaces 2 as it was used least recently. But again we required 2, page fault occurs and it replaces 3, since it is now least recently used of three pages in the memory. The process continues till end.

Page faults = 12 Page hits = 8

Advantages:

① Gives highest priority to least recently used page

② Better than FIFO

Disadvantages:

① Difficult to implement

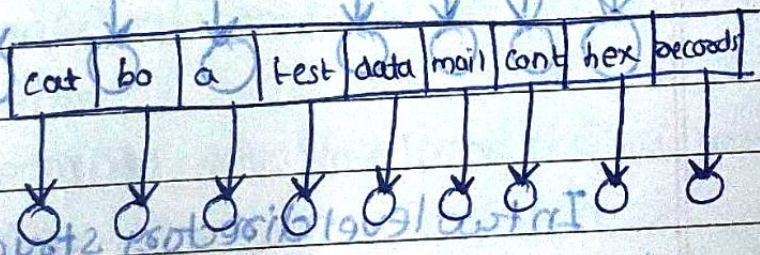
② LRU may replace pages that are just about to be used again.

Directory Structure :-

Directory structure is a collection of nodes containing information about all files, which are present in it. In directory structure, we need to consider following operations:

① Search for a file
② Create a file
③ Delete a file
④ List a directory
⑤ Rename a file
⑥ Traverse a file system

Single-level directory structure :-



In single level directory structure all files are contained in the same directory

Advantages :-

① It is simplest directory structure

② It is easy to support and understand.

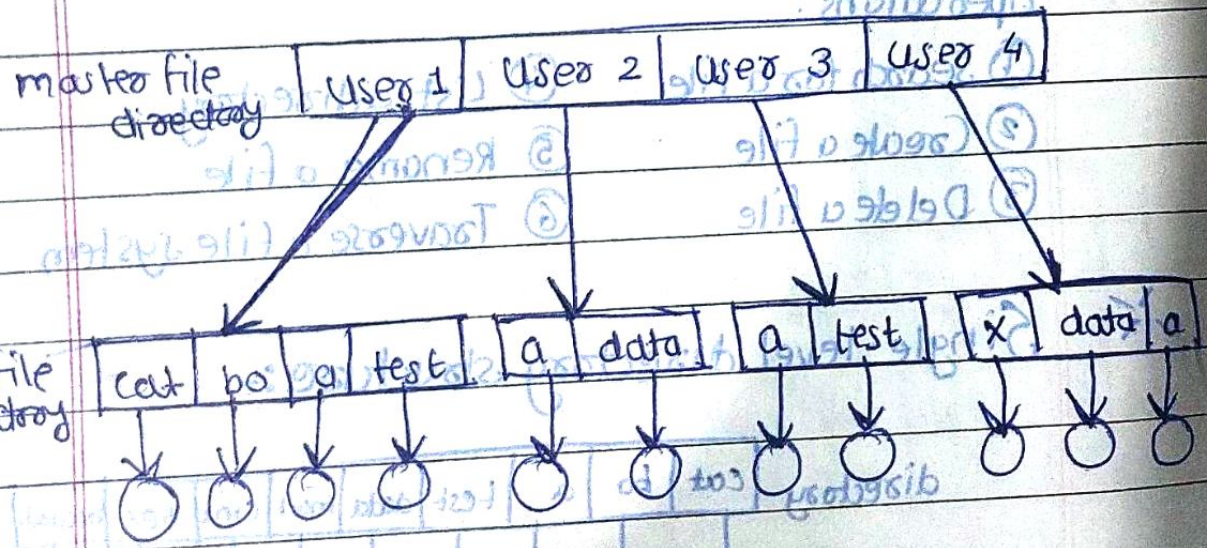
Disadvantages / Limitations :-

① When number of files and users increases, it is difficult to manage

② Since all files are in same directory, they must have unique names

③ Even a single user may find it difficult to remember all the names of the files as the number of files increase

Two level directory structure:



In two level directory structure, each user has has own user File Directory (UFD). When user job starts or a user logs in, the system's Master File Directory (MFD) is searched.

A two level directory is a tree of height 2. The root of the tree is MFD. Its direct descendants are the UFD's. The descendants of UFD's are files and files are the leaves of the tree.

While creating or deleting file for a user, the OS searches only that user's UFD. To name a particular file uniquely in a two level directory, we must give both user name and the file name. Thus, a user name and a file name define a path name.

Advantages:

1. Solves the name-collision problem
2. When user refers to the particular file, only his own UFD is searched. Thus different users may have files with the same name.
3. Isolate of one user from another is advantageous when the users are completely independent.

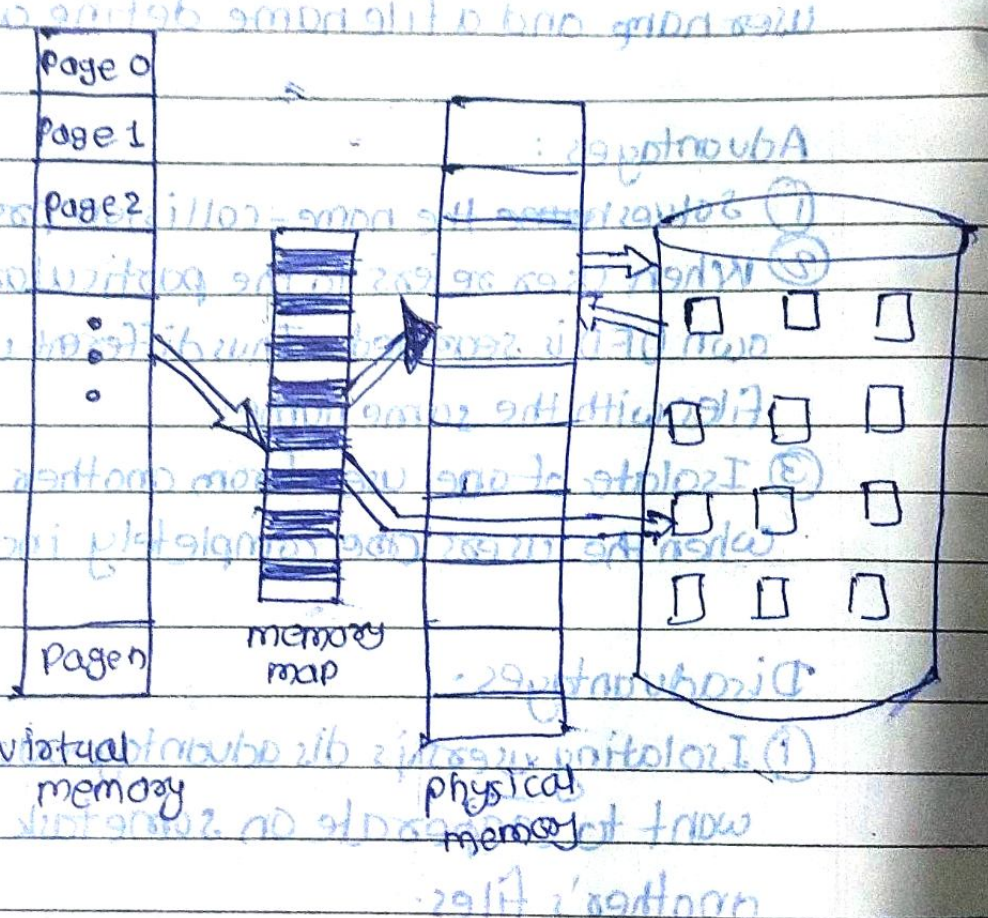
Disadvantages:

1. Isolating users is disadvantageous when the users want to cooperate on some task and to access one another's files.
2. Some systems do not allow local user files to be accessed by other users.

Page No. 52
Date

Virtual Memory :

Virtual memory is a memory that appears to exist as main storage although most of it is supported by data held in secondary storage, transfer between the two being made automatically as required. It is a technique that allows execution of processes that are not completely in the main memory.



In many real programs, the entire program is not needed at a time. So we can keep only necessary part of the program in the main memory and the rest of the part can be kept on virtual memory.

Advantages :-

① Due to virtual memory, programs can be larger than physical memory.
② Virtual memory abstracts main memory into very large, uniform array of storage.
③ It seperates logical memory from physical memory.
④ It frees programmers from memory-storage limitations.
⑤ It allows processes to share files easily and to implement shared memory.
⑥ It provides efficient mechanism for process creation.
⑦ It increases CPU utilization and efficiency.

Disadvantages :-

① It is not easy to implement.
② It may decrease performance if it is not used carefully.
③ Its complexity and cost may be higher.

Segmentation :-

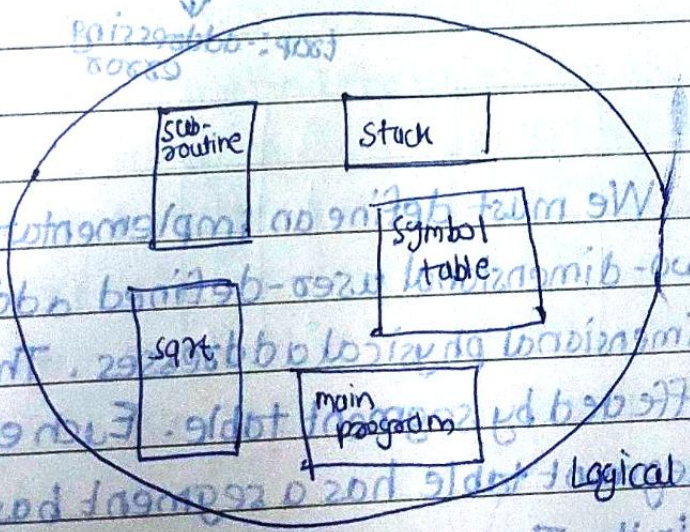
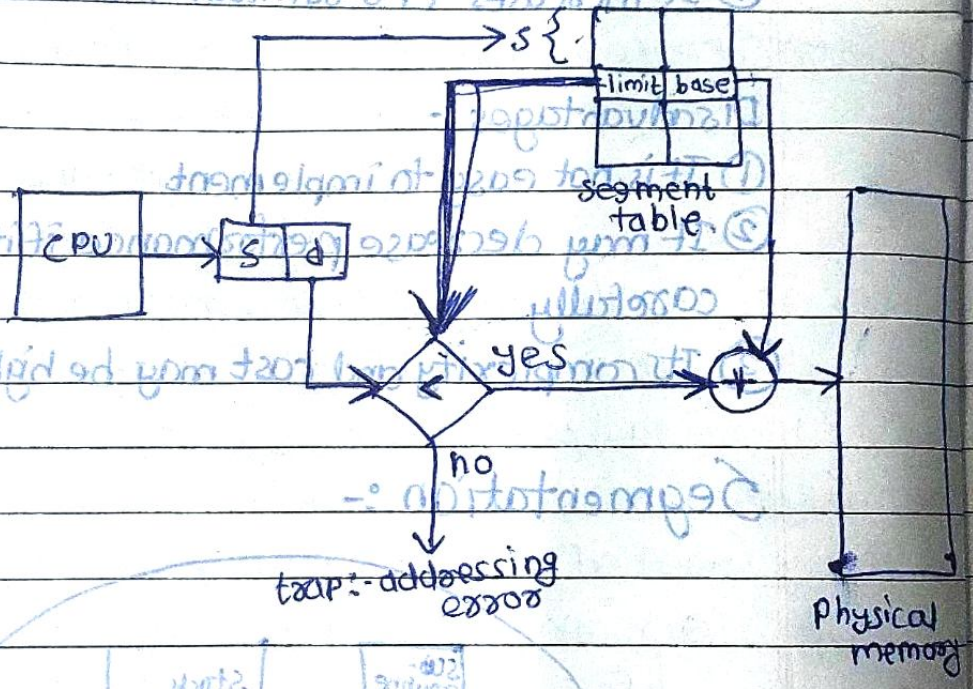


Fig: Programmer's view of a program

Segmentation is a memory management scheme that supports the programmer's view of memory. A logical address space is a collection of segments. Each segment has a name and a length. The addresses specify both the segment name and the offset within the segment.

Thus a logical address consist of a two tuple
<segment number, offset>

Segmentation Hardware :-



We must define an implementation to map two-dimensional user-defined addresses into one dimensional physical addresses. The mapping is effected by segment table. Each entry in the segment table has a segment base and a segment limit. The segment base contains the starting physical address where the segment resides in memory and segment limit specifies the length of segment.

A logical address consists of two parts :-

① A segment no. (s)

② An offset into that segment (d)

The segment number is used as index to the segment table. The offset d of logical address must be between 0 and the segment limit. If it is not, we trap to the operating system.

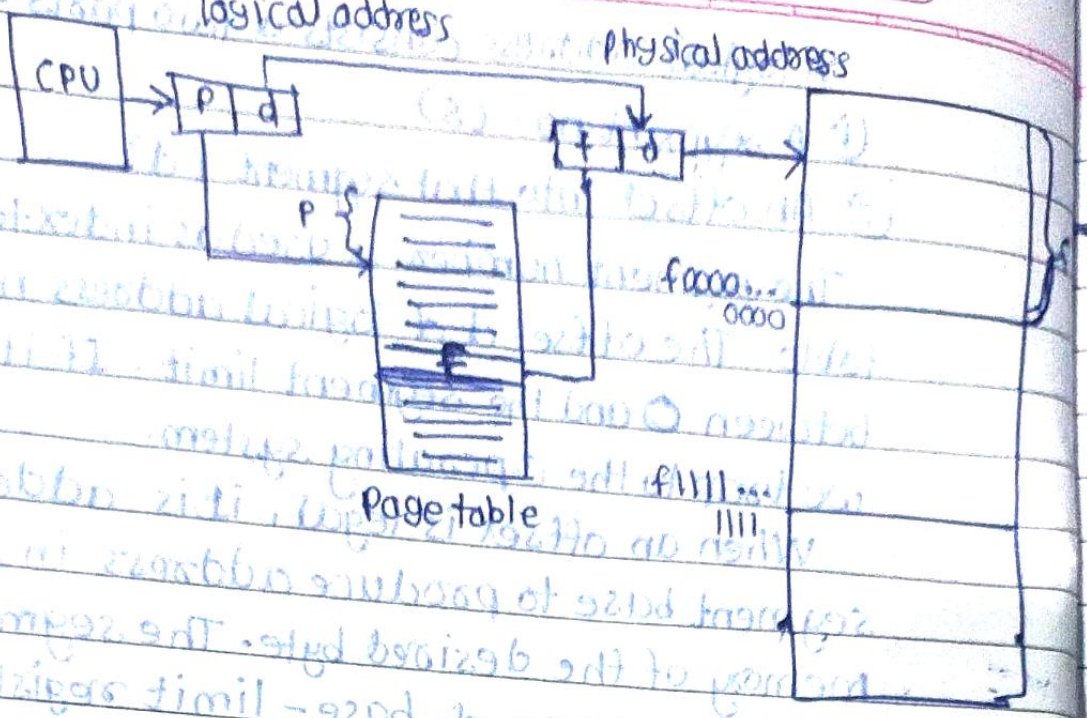
When an offset is legal, it is added to the segment base to produce address in physical memory of the desired byte. The segment table is thus an array of base-limit register pairs.

Paging :- Paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in memory. In this scheme, the OS retrieves data from secondary storage in same size blocks called pages.

Basic method :-

The basic method for implementing paging involves breaking physical memory into fixed sized blocks called frames and breaking logical memory into blocks of same size called pages.

When a process is to be executed, its pages are loaded into available memory frames from their source.



Every address generated by CPU is divided into two parts

① A page number (p)

② A page offset (d)

A page number is used as an index to a page table. The page table contains the base address each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.

		frame no.	
Page 0	0	1	0
Page 1	1	4	1 Page 0
Page 2	2	3	2
Page 3	3	7	3 Page 2
			4 Page 1
			5
			6
			7 Page 3

logical memory

page table

physical memory

Date

Difference between Paging & Segmentation:

	Paging	Segmentation
①	Pages are used for swapping or managing memory	Small pieces called segments are used for memory management
②	Page is indicated by its number and offset	segment is indicated by segment number & its offset
③	Page table is formed	segment table is formed
④	Do not support user's view of memory	Supports user's view of memory
⑤	Page frames are the physical units used in paging	The big blocks are formed in the segmentation
⑥	Paging algorithms are complicated than segmentation	segmentation algorithms are relatively easy to implement