Scheme of Evaluation

Internal Assessment Test 3– May 2024

Sub:	Operating System Concepts							Sub Code:	22MCA12
Date:	20-05-24	Duration:	90 mins	Max Marks:	50	Sem:	I	Branch:	MCA

Q.NO	Description	Marks Distribution	Max Marks
1	**Write notes on i) Dynamic Loading ii) Dynamic Linking.** • Explanation of Dynamic Loading • Explanation of Dynamic Linking	 5 5	10
2	**Differentiate between internal and external fragmentation.** • Give minimum 6 list out the points	 10	10
3	**What is Paging? Explain the paging hardware with a neat diagram.** • Definition of Paging • Explanation of Paging hardware • Draw the diagram	 2 6 2	10
4	**What is the need for segmentation? Explain the segmentation architecture with a neat diagram.** • Definition of Segmentation • Explanation of segmentation architecture • Draw the diagram	 2 6 2	10
5	**What is demand paging? Explain with a neat diagram.** • Definition of demand paging • Explanation of demand paging with neat diagram	 2 8	10
6	**Consider the following reference string:** **7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1** **How many page faults would occur in case ofi)FIFO ii)Optimal algorithm?** • Solution of FIFO algorithm • Solution of Optimal Algorithm	 5 5	10
7	**What are the various access methods used to access files?** • Explanation of access methods	 10	10

8	**Explain the different directory structures with neat diagrams.** • Explanation of types of directory structures • Illustrate with diagram for each directory structure	**7** **3**	**10**
9	**Explain how free space management is done using i) Bit Vector ii) Linked List iii) Grouping and iv) Counting.** • Explanation of) Bit Vector ii) Linked List iii) Grouping and iv) Counting. • Illustrate the diagram	**8** **2**	**10**
10	**Explain the file allocation methods with neat diagrams.** • Explanation of types of allocation methods	**10**	**10**

Internal Assessment Test 3 – May 2024

Operating System Concepts							Sub Code:	22MCA12
20/04/2024	Duration:	90 min's	Max Marks:	50	Sem:	I	Branch:	MCA

PART I

1. **Write notes on i) Dynamic Loading ii) Dynamic Linking.**

Dynamic Loading

- It loads the program and data dynamically into physical memory to obtain better memory-space utilization.
- With dynamic loading, a routine is not loaded until it is called.
- Dynamic loading does not require special support from the operating system.
- Dynamic loading is used to obtain better memory utilization.
- In dynamic loading the routine or procedure will not be loaded until it is called.
- Whenever a routine is called, the calling routine first checks whether the called routine is already loaded or not. If it is not loaded it cause the loader to load the desired program in to the memory and updates the programs address table to indicate the change and control is passed to newly called routine.

Advantage:

Gives better memory utilization.

- Unused routine is never loaded.
- Do not need special operating system support.
- This method is useful when large amount of codes are needed to handle in frequently occurring cases.

Dynamic Linking

- Linking postponed until execution time.
- Small piece of code (stub) used to locate the appropriate memory-resident library routine.
- Stub replaces itself with the address of the routine and executes the routine.
- Operating system needed to check if routine is in processes memory address.
- Dynamic linking is particularly useful for libraries.
- For a process to be executed it should be loaded in to the physical memory. The size of the process is limited to the size of the physical memory.

Dynamic linking and Shared libraries:

- Some operating system supports only the static linking.
- In dynamic linking only the main program is loaded in to the memory. If the main program requests a procedure, the procedure is loaded and the link is established at the time of references. This linking is postponed until the execution time.
- With dynamic linking a "stub" is used in the image of each library referenced routine.
- A "stub" is a piece of code which is used to indicate how to locate the appropriate memory resident library routine or how to load library if the routine is not already present. x When "stub" is executed it checks whether the routine is present is memory or not. If not it loads the routine in to the memory.
- This feature can be used to update libraries i.e., library is replaced by a new version and all the programs can make use of this library.

- More than one version of the library can be loaded in memory at a time and each program uses its version of the library. Only the program that are compiled with the new version are affected by the changes incorporated in it.
- Other programs linked before new version is installed will continue using older libraries this type of system is called "shared library".

2. Differentiate between internal and external fragmentation.

S.No	Internal fragmentation	External fragmentation
1.	In internal fragmentation fixed-sized memory, blocks square measure appointed to process.	In external fragmentation, variable-sized memory blocks square measure appointed to the method.
2.	Internal fragmentation happens when the method or process is smaller than the memory.	External fragmentation happens when the method or process is removed.
3.	The solution of internal fragmentation is the best-fit block.	The solution to external fragmentation is compaction and paging.
4.	Internal fragmentation occurs when memory is divided into fixed-sized partitions.	External fragmentation occurs when memory is divided into variable size partitions based on the size of processes.
5.	The difference between memory allocated and required space or memory is called Internal fragmentation.	The unused spaces formed between non-contiguous memory fragments are too small to serve a new process, which is called External fragmentation.
6.	Internal fragmentation occurs with paging and fixed partitioning.	External fragmentation occurs with segmentation and dynamic partitioning.
7.	It occurs on the allocation of a process to a partition greater than the process's requirement. The leftover space causes degradation system performance.	It occurs on the allocation of a process to a partition greater which is exactly the same memory space as it is required.
8.	It occurs in worst fit memory allocation method.	It occurs in best fit and first fit memory allocation method.

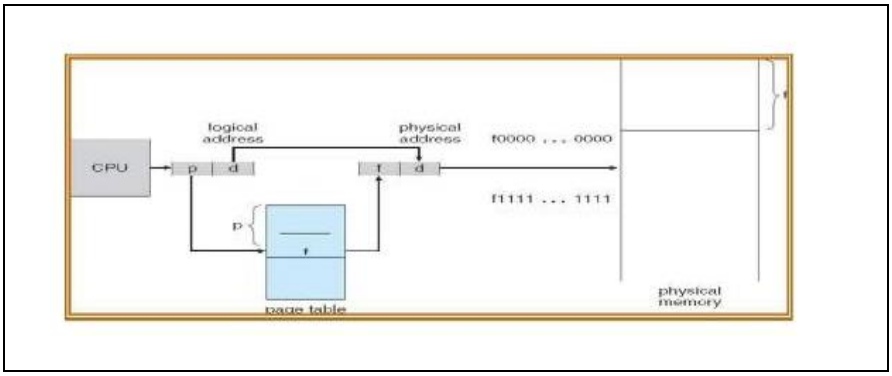
PART II

3. What is paging? Explain the paging hardware with a neat diagram.

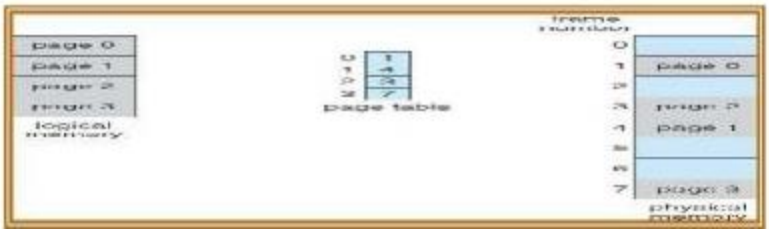
- Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous. Support for paging is handled by hardware.
- It is used to avoid external fragmentation.
- Paging avoids the considerable problem of fitting the varying sized memory chunks on to the backing store.
- When some code or date residing in main memory need to be swapped out, space must be found on backing store.

Basic Method:

- Physical memory is broken in to fixed sized blocks called frames (f).
- Logical memory is broken in to blocks of same size called pages (p).
- When a process is to be executed its pages are loaded in to available frames from backing store.
- The blocking store is also divided in to fixed-sized blocks of same size as memory frames.
- The following figure shows paging hardware:



- Logical address generated by the CPU is divided in to two parts: page number (p) and page offset (d).
- The page number (p) is used as index to the page table. The page table contains base address of each page in physical memory. This base address is combined with the page offset to define the physical memory i.e., sent to the memory unit.
- The page size is defined by the hardware. The size of a power of 2, varying between 512 bytes and 10Mb per page.
- If the size of logical address space is 2^m address unit and page size is 2^n, then high order m-n designates the page number and n low order bits represents page offset



Eg:-To show how to map logical memory in to physical memory consider a page size of 4 bytes and physical memory of 32 bytes (8 pages).

a. Logical address 0 is page 0 and offset 0. Page 0 is in frame 5. The logical address 0 maps to physical address 20. [(5*4) + 0].
b. Logical address 3 is page 0 and offset 3 maps to physical address 23 [(5*4) + 3].

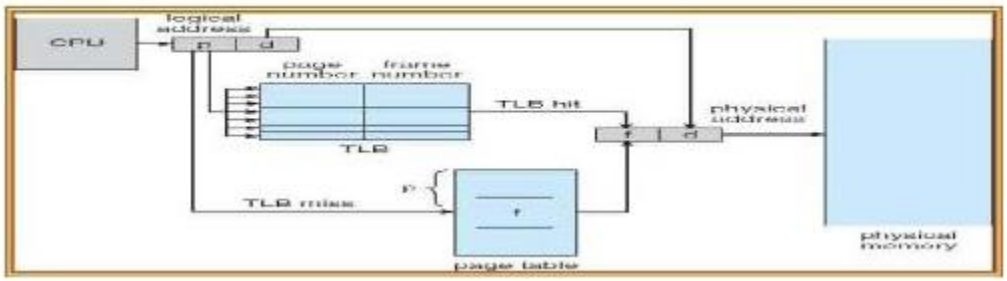
Hardwar e Support for Pag i ng:

The hardware implementation of the page table can be done in several ways:

1. The simplest method is that the page table is implemented as a set of dedicated registers. These registers must be built with very high speed logic for making paging address translation. Every accessed memory must go through paging map. The use of registers for page table is satisfactory if the page table is small.

2. If the page table is large then the use of registers is not visible. So the page table is kept in the main memory and a page table base register [PTBR] points to the page table. Changing the page table requires only one register which reduces the context switching type. The problem with this approach is the time required to access memory location. To access a location [i] first we have to index the page table using PTBR offset. It gives the frame number which is combined with the page offset to produce the actual address. Thus we need two memory accesses for a byte.

3. The only solution is to use special, fast, lookup hardware cache called translation look a side buffer[TLB]or associative register. TLB is built with associative register with high speed memory. Each register contains two paths a ke y and a value.



When an associative register is presented with an item, it is compared with all the key values, if found the corresponding value field is return and searching is fast. TLB is used with the page table as follows:

- TLB contains only few page table entries.
- When a logical address is generated by the CPU, its page number along with the frame number is added to TLB. If the page number is found its frame memory is used to access the actual memory.
- If the page number is not in the TLB (TLB miss) the memory reference to the page table is made. When the frame number is obtained use can use it to access the memory.
- If the TLB is full of entries the OS must select anyone for replacement.
- Each time a new page table is selected the TLB must be flushed [erased] to ensure that next executing process do not use wrong information.
- The percentage of time that a page number is found in the TLB is called HIT ratio.

4. **What is the need for segmentation? Explain the segmentation architecture with a neat diagram.**

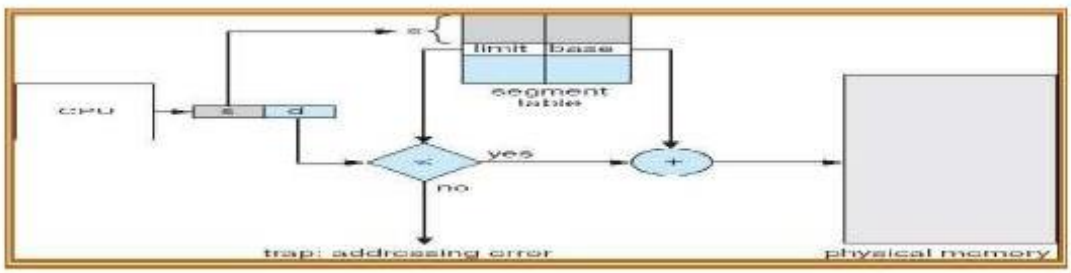
Basic method:

- Most users do not think memory as a linear array of bytes rather the users thinks memory as a collection of variable sized segments which are dedicated to a particular use such as code, data, stack, heap etc.
- A logical address is a collection of segments. Each segment has a name and length. The address specifies both the segment name and the offset within the segments.
- The user specifies address by using two quantities: a segment name and an offset.
- For simplicity the segments are numbered and referred by a segment number. So the logical address consists of <segment number, offset>.

Hardware support:

We must define an implementation to map 2D user defined address in to 1D physical address.

This mapping is affected by a segment table. Each entry in the segment table has a segment base and segment limit. The segment base contains the starting physical address where the segment resides and limit specifies the length of the segment.



The use of segment table is shown in the above figure:

- Logical address consists of two parts: segment number‘s’ and an offset‘d’ to that segment.
- The segment number is used as an index to segment table.
- The offset ’d’ must bi in between 0 and limit, if not an error is reported to OS.
- If legal the offset is added to the base to generate the actual physical address.
- The segment table is an array of base limit register pairs.

Protection and Shar ing:

- A particular advantage of segmentation is the association of protection with the segments.
- The memory mapping hardware will check the protection bits associated with each segment table entry to prevent illegal access to memory like attempts to write in to read-only segment.
- Another advantage of segmentation involves the sharing of code or data. Each process has a segment table associated with it. Segments are shared when the entries in the segment tables of two different processes points to same physical location.
- Sharing occurs at the segment table. Any information can be shared at the segment level. Several segments can be shared so a program consisting of several segments can be shared.

We can also share parts of a program.

Advantages:

- Eliminates fragmentation.
- Provides virtual growth.
- Allows dynamic segment growth
- Assist dynamic linking.
- Segmentation is visible

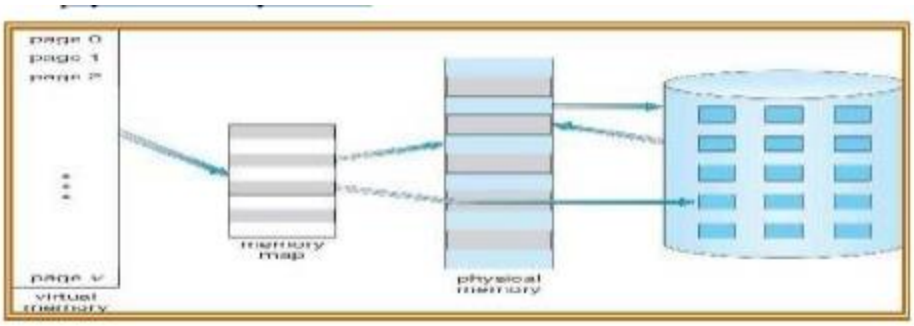
PART III

5. What is demand paging? Explain with a neat diagram.

A demand paging is similar to paging system with swapping when we want to execute a process we swap the process the in to memory otherwise it will not be loaded in to memory.

A swapper manipulates the entire processes, where as a pager manipulates individual pages of the process.

Basic concept: Instead of swapping the whole process the pager swaps only the necessary pages in to memory. Thus it avoids reading unused pages and decreases the swap time and amount of physical memory needed.



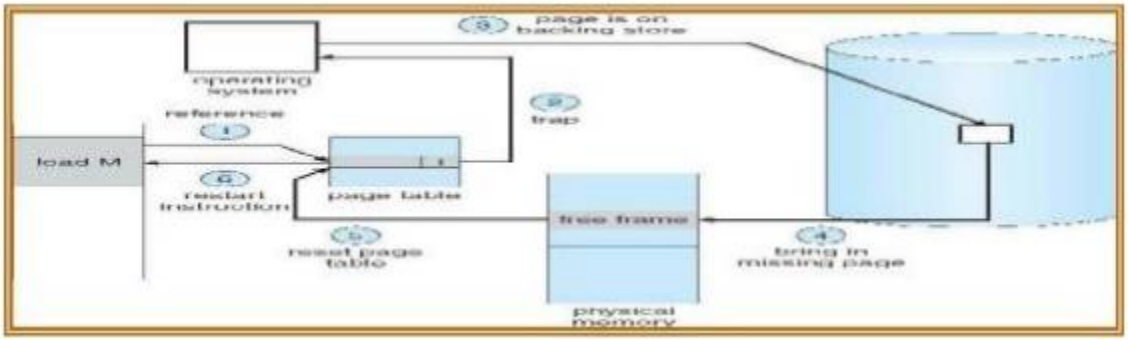
The valid-invalid bit scheme can be used to distinguish between the pages that are on the disk and that are in memory.

• If the bit is valid then the page is both legal and is in memory.

•If the bit is invalid then either page is not valid or is valid but is currently on the disk. Marking a page as invalid will have no effect if the processes never access to that page. Suppose if it access the page which is marked invalid, causes a page fault trap. This may result in failure of OS to bring the desired page in to memory.

The step for handling page fault is straight forward and is given below:

1. We check the internal table of the process to determine whether the reference made is valid or invalid.
2. If invalid terminate the process,. If valid, then the page is not yet loaded and we now page it in.
3. We find a free frame.
4. We schedule disk operation to read the desired page in to newly allocated frame.
5. When disk reed is complete, we modify the internal table kept with the process to indicate that the page is now in memory.
6. We restart the instruction which was interrupted by illegal address trap. The process can now access the page. In extreme cases, we start the process without pages in memory. When the OS points to the instruction of process it generates a page fault. After this page is brought in to memory the process continues to execute, faulting as necessary until every demand paging i.e., it never brings the page in to memory until it is required.



Hardware support: For demand paging the same hardware is required as paging and swapping.
1. Page table:-Has the ability to mark an entry invalid through valid-invalid bit.
2. Secondary memory:-This holds the pages that are not present in main memory.
Performance of demand paging:
Demand paging can have significant effect on the performance of the computer system. Let P be the probability of the page fault (0<=P<=1) Effective access time = (1-P) * ma + P * page fault. Where P = page fault and ma = memory access time. Effective access time is directly proportional to page fault rate. It is important to keep page fault rate low in demand paging

A page fault causes the following sequence to occur:

- Trap to the OS. Save the user registers and process state.
- Determine that the interrupt was a page fault.
- Checks the page references were legal and determine the location of page on disk. Issue a read from disk to a free frame.
- If waiting, allocate the CPU to some other user.
- Interrupt from the disk. Save the registers and process states of other users.
- Determine that the interrupt was from the disk.
- Correct the page table and other table to show that the desired page is now in memory.
- Wait for the CPU to be allocated to this process again.
- Restore the user register process state and new page table then resume the interrupted instruction.

6. **Consider the following reference string:**
   7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
   How many page faults would occur in case of i)FIFO ii)Optimal algorithm?

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

In FIFO Algorithm 15 page fault will occur

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

In Optimal Algorithm 9 page fault will occur

PART IV

7. What are the various access methods used to access files?

Access Methods

The file information is accessed and read into computer memory. The information in the file can be accessed in several ways.

a) Sequential Access

- Here information in the file is processed in order, one record after the other.
- This mode of access is a common method; for example, editors and compilers usually access files in this fashion.
- A sequential access file emulates magnetic tape operation, and generally supports a few operations: o read next - read a record and advance the file pointer to the next position. o write next - write a record to the end of file and advance the file pointer to the next position. o skip n records - May or may not be supported. 'n' may be limited to positive numbers, or may be limited to +/- 1.

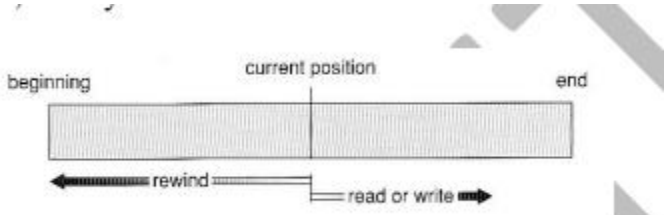


Figure 10.3 Sequential-access file.

b) Direct Access

A file is made up of fixed-length logical records that allow programs to read and write records randomly. The records can be rapidly accessed in any order.

Direct access are of great use for immediate access to large amount of information. Eg : Database file. When a query occurs, the query is computed and only the selected rows are access directly to provide the desired information. Operations supported include:

- read n - read record number n. (position the cursor to n and then read the record)
- write n - write record number n. (position the cursor to n and then write the record)
- jump to record n – move to nth record (n- could be 0 or the end of file)
- If the record length is L, there is a request for record 'N'. Then the direct access to the starting byte of record 'N' is at L*(N-1) Eg: if 3rd record is required and length of each record(L) is 50, then the starting position of 3rd record is L*(N-1) Address = 50*(3-1) = 100.

c) Other Access Methods (Indexed method)

- These methods generally involve the construction of an index for the file called index file.
- The index file is like an index page of a book, which contains key and address. To find a record in the file, we first search the index and then use the pointer to access the record directly and find the desired record.
- An indexed access scheme can be easily built on top of a direct access system.

- For very large files, the index file itself is very large. The solution to this is to create an index for index file. i.e. multi-level indexing.

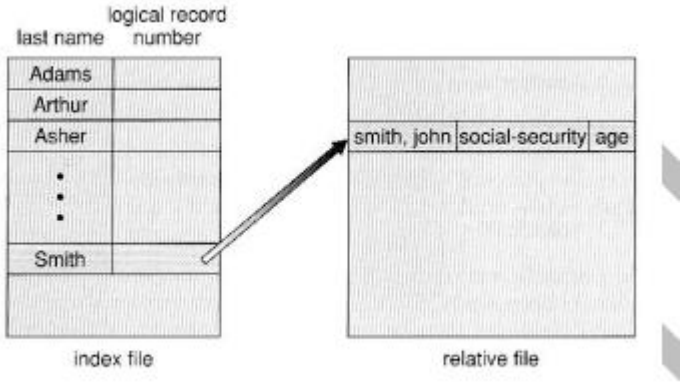


Figure 10.5 Example of index and relative files.

8. Explain the different directory structures with neat diagrams.

Directory Structures –

a) Single-Level Directory

• It is the simplest directory structure.

• All files are contained in the same directory, which is easy to support and understand. The limitations of this structure is that –

• All files are in the same directory must have unique names.

• Even a single user on a single-level directory may find it difficult to remember the names of all the files as the number of files increases.

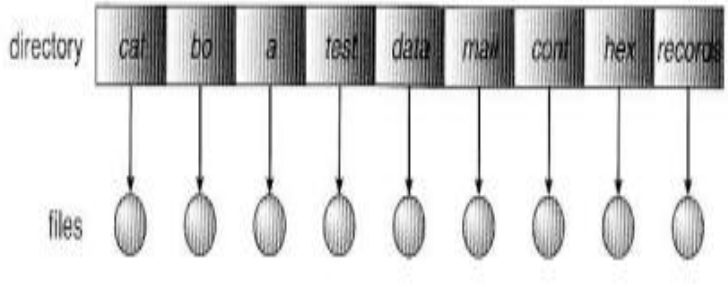


Figure 10.7 Single-level directory.

b) Two-Level Directory

• Each user gets their own directory space - user file directory(UFD)

• File names only need to be unique within a given user's directory.

• A master file directory(MFD) is used to keep track of each users directory, and must be maintained when users are added to or removed from the system.
• When a user refers to a particular file, only his own UFD is searched.
• All the files within each UFD are unique.
• To create a file for a user, the operating system searches only that user's UFD to ascertain whether another file of that name exists.

- To delete a file, the operating system confines its search to the local UFD; thus, it cannot accidentally delete another user's file that has the same name. The user directories themselves must be created and deleted as necessary.
- This structure isolates one user from another. Isolation is an advantage when the users are completely independent but is a disadvantage when the users want to cooperate on some task and to access one another's files

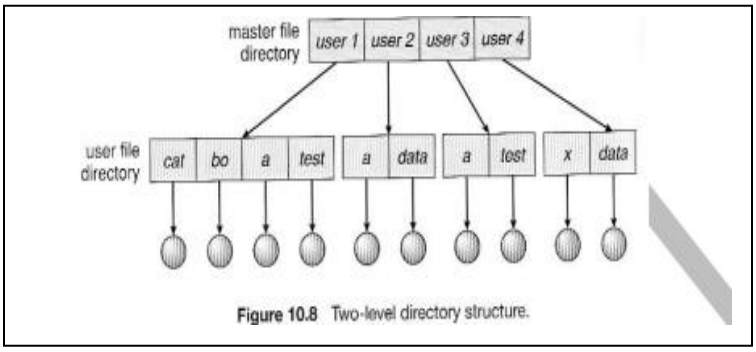


Figure 10.8 Two-level directory structure.

.

c) Tree-Structured Directories

• A tree structure is the most common directory structure.
• The tree has a root directory, and every file in the system has a unique path name.
• A directory (or subdirectory) contains a set of files or subdirectories.
• One bit in each directory entry defines the entry as a file (0) or as a subdirectory (1). Special system calls are used to create and delete directories.
• Path names can be of two types: absolute and relative. An absolute path begins at the root and follows a down to the specified file, giving the directory names on the path. A relative path defines a path from the current directory.
• For example, in the tree-structured file system of figure below if the current directory is root/spell/mail, then the relative path name is prt/first and the files absolute path name root/spell/mail/prt/jirst.

- Directories are stored the same as any other file in the system, except there is a bit that identifies them as directories, and they have some special structure that the OS understands.

• One question for consideration is whether or not to allow the removal of directories that are not empty - Windows requires that directories be emptied first, and UNIX provides an option for deleting entire sub-trees.

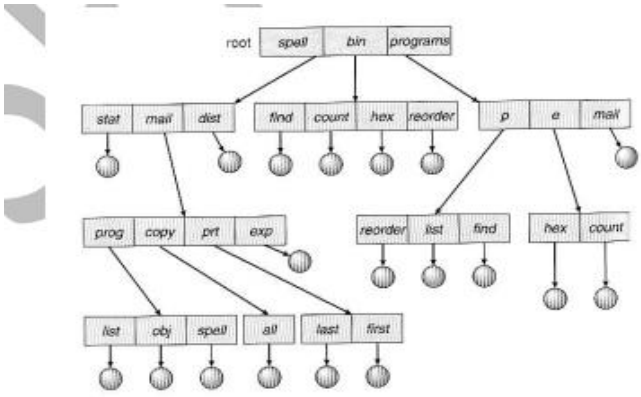


Figure 10.9 Tree-structured directory structure.

d) Acyclic-Graph Directories

• When the same files need to be accessed in more than one place in the directory structure (e.g. because they are being shared by more than one user), it can be useful to provide an acyclic-graph structure. (Note the directed arcs from parent to child.)

- UNIX provides two types of links (pointer to another file)for implementing the acyclic-graph structure.
- A hard link (usually just called a link) involves multiple directory entries that both refer to the same file. Hard links are only valid for ordinary files in the same filesystem.
- A symbolic link, that involves a special file, containing information about where to find the linked file. Symbolic links may be used to link directories and/or files in other filesystems, as well as ordinary files in the current filesystem.
- Windows only supports symbolic links, termed shortcuts.
- Hard links require a reference count, or link count for each file, keeping track of how many directory entries are currently referring to this file. Whenever one of the references is removed the link count is reduced, and when it reaches zero, the disk space can be reclaimed.

For symbolic links there is some question as to what to do with the symbolic links when the original file is moved or deleted: o One option is to find all the symbolic links and adjust them also.

Another is to leave the symbolic links dangling, and discover that they are no longer valid the next time they are used. o What if the original file is removed, and replaced with another file having the same name before the symbolic link is next used? Another approach to deletion is to preserve the file until all references to it are deleted. To implement this approach, we must have some mechanism for determining that the last reference to the file has been deleted. When a link or a copy of the directory entry is established, a new entry is added to the filereference list. When a link or directory entry is deleted, we remove its entry on the list. The file is deleted when its file-reference list is empty.

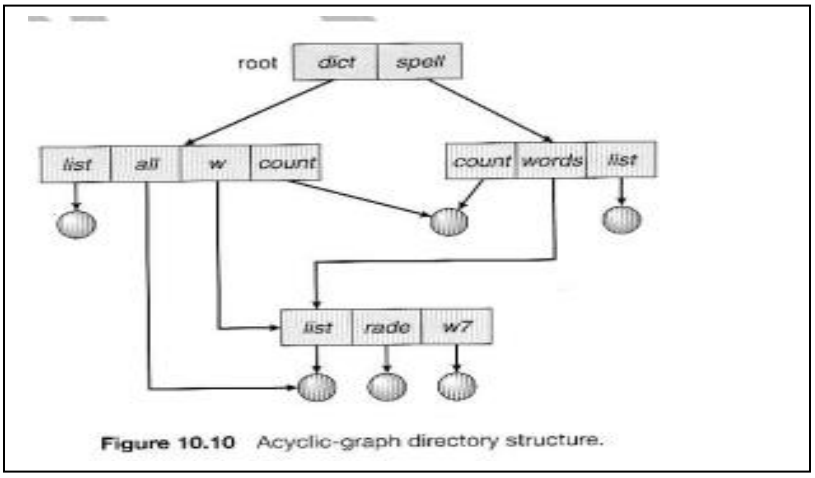


Figure 10.10 Acyclic-graph directory structure.

e) General Graph Directory

• If cycles are allowed in the graphs, then several problems can arise: o Search algorithms can go into infinite loops.

One solution is to not follow links in search algorithms. (Or not to follow symbolic links, and to only allow symbolic links to refer to directories)

- Sub-trees can become disconnected from the rest of the tree and still not have their reference counts reduced to zero. Periodic garbage collection is required to detect and resolve this problem. (chkdsk in DOS and fsck in UNIX search for these problems, among others, even though cycles are not supposed to be allowed in either system. Disconnected disk blocks that are not marked as free are added back to the file systems with made-up file names, and can usually be safely deleted.)

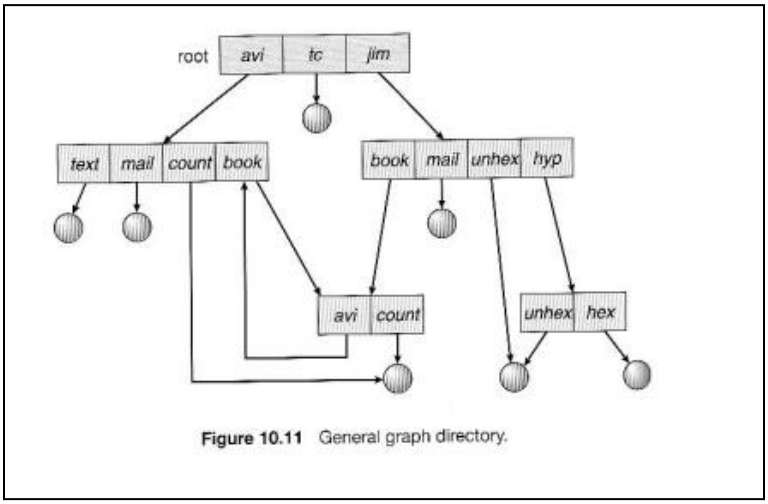


Figure 10.11 General graph directory.

PART V

9. Explain how free space management is done using i) Bit Vector ii) Linked List iii) Grouping and iv) Counting.

Free space management is a critical aspect of operating systems as it involves managing the available storage space on the hard disk or other secondary storage devices. The operating system uses various techniques to manage free space and optimize the use of storage devices.

The system keeps tracks of the free disk blocks for allocating space to files when they are created. Also, to reuse the space released from deleting the files, free space management becomes crucial. The system maintains a free space list which keeps track of the disk blocks that are not allocated to some file or directory. The free space list can be implemented mainly as:

1. **Bitmap or Bit vector –** A Bitmap or Bit Vector is series or collection of bits where each bit corresponds to a disk block. The bit can take two values: 0 and 1: 0 indicates that the block is allocated and 1 indicates a free block. The given instance of disk blocks on the disk in Figure 1 (where green blocks are allocated) can be represented by a bitmap of 16 bits as: 0000111000000110.

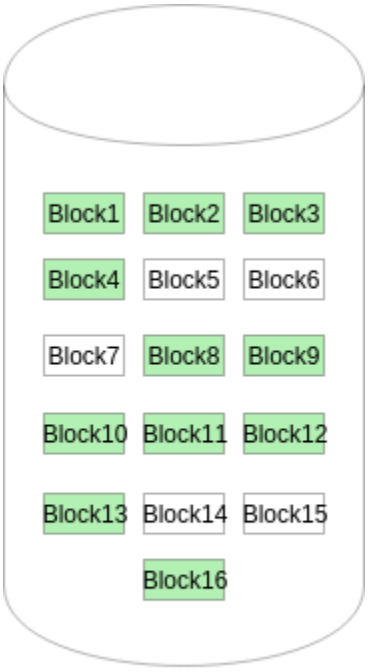


Figure - 1

Advantages –

- Simple to understand.
- Finding the first free block is efficient. It requires scanning the words (a group of 8 bits) in a bitmap for a non-zero word. (A 0-valued word has all bits 0). The first free block is then found by scanning for the first 1 bit in the non-zero word.

2. **Linked List** – In this approach, the free disk blocks are linked together i.e. a free block contains a pointer to the next free block. The block number of the very first disk block is stored at a separate location on disk and is also coached in memory.

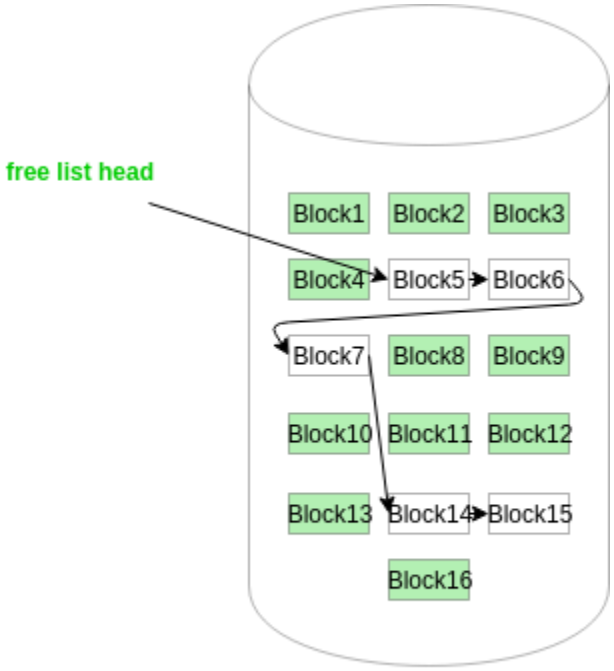


Figure - 2

In Figure-2, the free space list head points to Block 5 which points to Block 6, the next free block and so on. The last free block would contain a null pointer indicating the end of free list. A drawback of this method is the I/O required for free space list traversal.

3. **Grouping –** This approach stores the address of the free blocks in the first free block. The first free block stores the address of some, say n free blocks. Out of these n blocks, the first n-1 blocks are actually free and the last block contains the address of next free n blocks. An advantage of this approach is that the addresses of a group of free disk blocks can be found easily.
4. **Counting –** This approach stores the address of the first free disk block and a number n of free contiguous disk blocks that follow the first block. Every entry in the list would contain:

   Address of first free disk block

   A number n

   Here are some advantages and disadvantages of free space management techniques in operating systems:

   Advantages:

   - Efficient use of storage space: Free space management techniques help to optimize the use of storage space on the hard disk or other secondary storage devices.

- Easy to implement: Some techniques, such as linked allocation, are simple to implement and require less overhead in terms of processing and memory resources.
- Faster access to files: Techniques such as contiguous allocation can help to reduce disk fragmentation and improve access time to files.

Disadvantages:

- Fragmentation: Techniques such as linked allocation can lead to fragmentation of disk space, which can decrease the efficiency of storage devices.
- Overhead: Some techniques, such as indexed allocation, require additional overhead in terms of memory and processing resources to maintain index blocks.
- Limited scalability: Some techniques, such as FAT, have limited scalability in terms of the number of files that can be stored on the disk.
- Risk of data loss: In some cases, such as with contiguous allocation, if a file becomes corrupted or damaged, it may be difficult to recover the data.
- Overall, the choice of free space management technique depends on the specific requirements of the operating system and the storage devices being used. While some techniques may offer advantages in terms of efficiency and speed, they may also have limitations and drawbacks that need to be considered.

10. Explain the file allocation methods with neat diagrams.

The allocation methods define how the files are stored in the disk blocks. There are three main disk space or file allocation methods.

- Contiguous Allocation
- Linked Allocation
- Indexed Allocation

The main idea behind these methods is to provide:

- Efficient disk space utilization.
- Fast access to the file blocks.

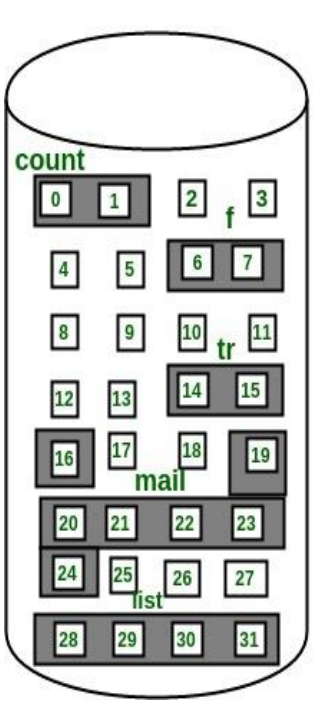
All the three methods have their own advantages and disadvantages as discussed below:

1. Contiguous Allocation

In this scheme, each file occupies a contiguous set of blocks on the disk. For example, if a file requires n blocks and is given a block b as the starting location, then the blocks assigned to the file will be: b, b+1, b+2,……b+n-1. This means that given the starting block address and the length of the file (in terms of blocks required), we can determine the blocks occupied by the file. The directory entry for a file with contiguous allocation contains

- Address of starting block
- Length of the allocated portion.

The file ‘mail’ in the following figure starts from the block 19 with length = 6 blocks. Therefore, it occupies 19, 20, 21, 22, 23, 24 blocks.



Directory

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

Advantages:

- Both the Sequential and Direct Accesses are supported by this. For direct access, the address of the kth block of the file which starts at block b can easily be obtained as (b+k).
- This is extremely fast since the number of seeks are minimal because of contiguous allocation of file blocks.

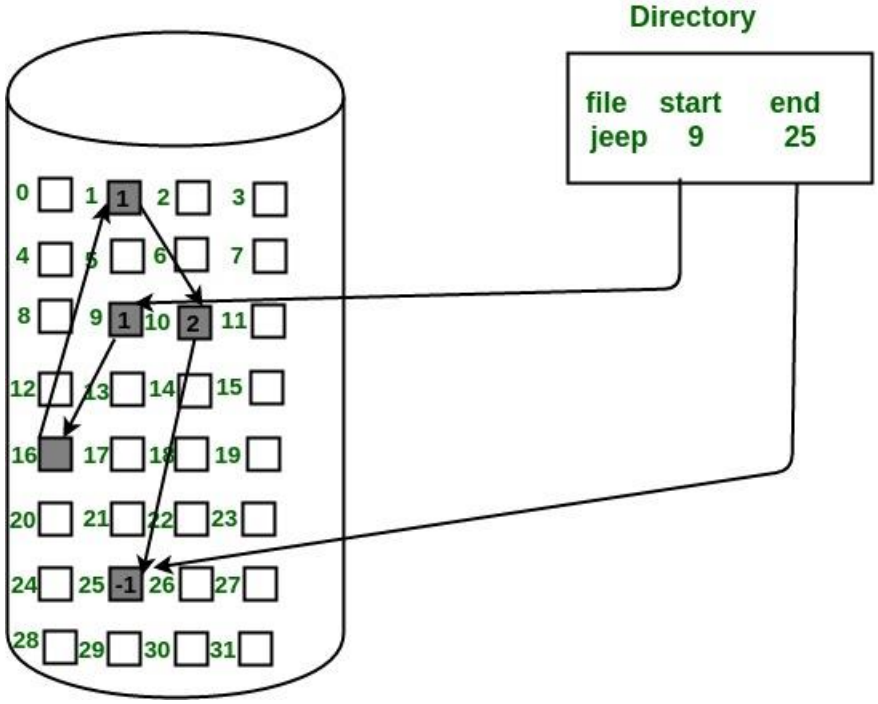
Disadvantages:

- This method suffers from both internal and external fragmentation. This makes it inefficient in terms of memory utilization.
- Increasing file size is difficult because it depends on the availability of contiguous memory at a particular instance.

2. Linked List Allocation

- In this scheme, each file is a linked list of disk blocks which need not be contiguous. The disk blocks can be scattered anywhere on the disk.
  The directory entry contains a pointer to the starting and the ending file block. Each block contains a pointer to the next block occupied by the file.

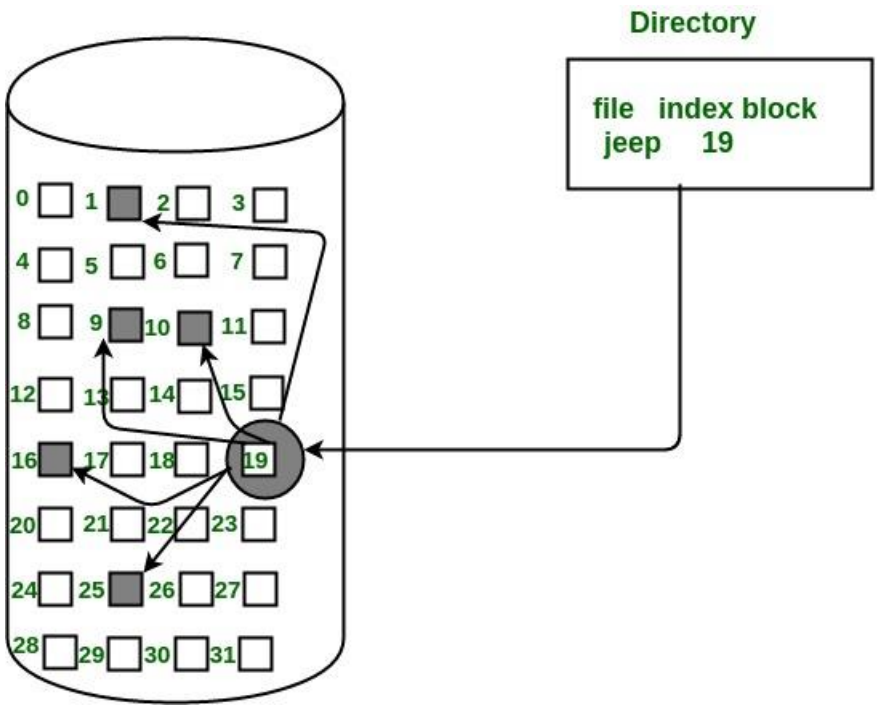
- The file ‘jeep’ in following image shows how the blocks are randomly distributed. The last block (25) contains -1 indicating a null pointer and does not point to any other block.



- Advantages:
- This is very flexible in terms of file size. File size can be increased easily since the system does not have to look for a contiguous chunk of memory.
- This method does not suffer from external fragmentation. This makes it relatively better in terms of memory utilization.
- Disadvantages:
- Because the file blocks are distributed randomly on the disk, a large number of seeks are needed to access every block individually. This makes linked allocation slower.
- It does not support random or direct access. We can not directly access the blocks of a file. A block k of a file can be accessed by traversing k blocks sequentially (sequential access) from the starting block of the file via block pointers.
- Pointers required in the linked allocation incur some extra overhead.

3. Indexed Allocation

- In this scheme, a special block known as the Index block contains the pointers to all the blocks occupied by a file. Each file has its own index block. The ith entry in the index block contains the disk address of the ith file block. The directory entry contains the address of the index block as shown in the image:



Advantages:

- This supports direct access to the blocks occupied by the file and therefore provides fast access to the file blocks.
- It overcomes the problem of external fragmentation.

Disadvantages:

- The pointer overhead for indexed allocation is greater than linked allocation.
- For very small files, say files that expand only 2-3 blocks, the indexed allocation would keep one entire block (index block) for the pointers which is inefficient in terms of memory utilization. However, in linked allocation we lose the space of only 1 pointer per block.