CMR INSTITUTE OF TECHNOLOGY, BENGALURU.
ACCREDITED WITH A+ GRADE BY NAAC

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

Course: Operating System

Code: 15CS64

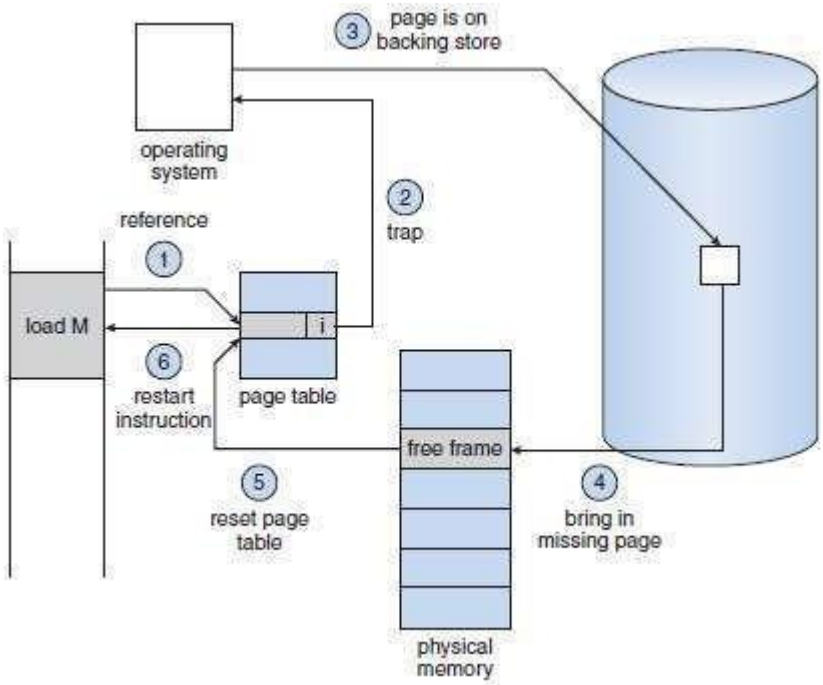
IAT3- Solution

1. What is a page fault? With a supporting diagram, explain the steps involved in handling page fault.

Solution:

- A **page-fault** occurs when the process tries to access a page that was not brought into memory**.**
- Procedure for handling the page-fault (Figure):

1) Check an internal-table to determine whether the reference was a valid or an invalid memory access.
2) If the reference is invalid, we terminate the process.
   If reference is valid, but we have not yet brought in that page, we now page it in.
3) Find a free-frame (by taking one from the free-frame list, for example).
4) Read the desired page into the newly allocated frame.
5) Modify the internal-table and the page-table to indicate that the page is now in memory.
6) Restart the instruction that was interrupted by the trap.



2. Consider the following reference string for a memory with 3 frames. How many page faults will occur for FIFO, LRU and optimal page replacement algorithms? Which is the most efficient algorithm? Reference string: 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1.

Solution:

(i) LRU with 3 frames:

Frames	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	4	4	4	0	0	0	1	1	1	1	1	1	1
2		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
3			1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
No. of Page faults	√	√	√	√		√		√	√	√	√			√		√		√		

No of page faults=12

(ii) FIFO with 3 frames:

Frames	7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
1	7	0	1	2	2	3	0	4	2	3	0	0	0	1	2	2	2	7	0	1
2		7	0	1	1	2	3	0	4	2	3	3	3	0	1	1	1	2	7	0
3			7	0	0	1	2	3	0	4	2	2	2	3	0	0	0	1	2	7
No. of Page faults	√	√	√	√		√	√	√	√	√	√			√	√			√	√	√

No of page faults=15

(iii) Optimal with 3 frames:

Frames	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	2	2	2	2	2	2	2	2	2	7	7	7
2		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
3			1	1	1	3	3	3	3	3	3	3	3	1	1	1	1	1	1	1
No. of Page faults	√	√	√	√		√		√			√			√				√		

No of page faults=9

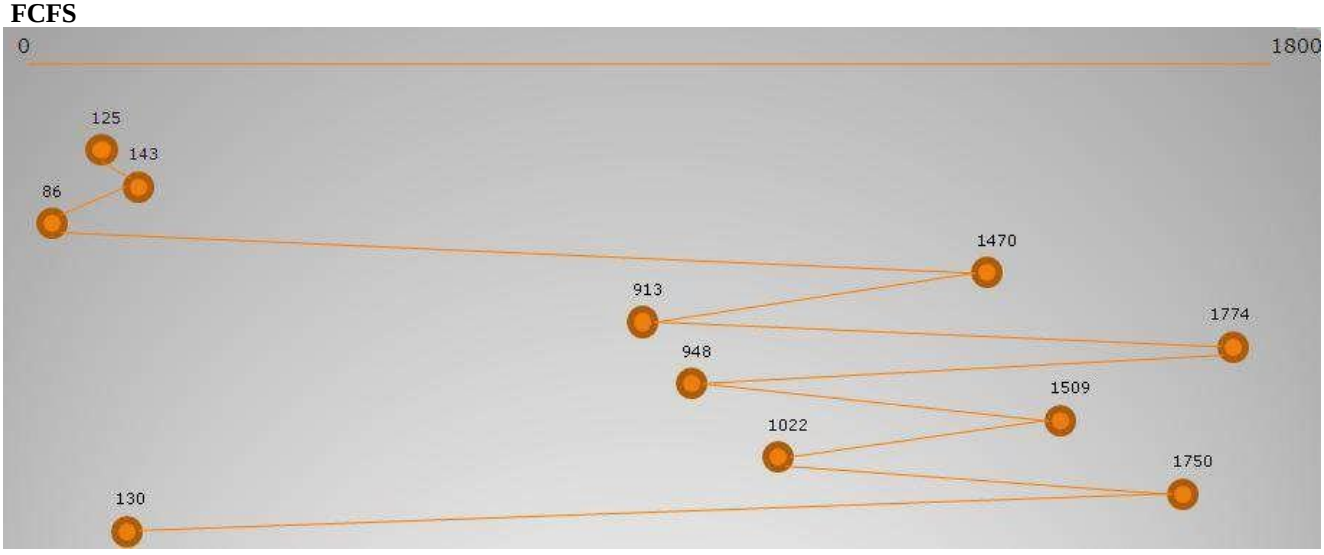
Conclusion: The optimal page replacement algorithm is most efficient among three algorithms, as it has lowest page faults i.e. 9.

3. A disk drive has 5000 cylinders numbered from 0 to 4999. If the disk head is currently at a cylinder 143, and a queue of pending requests are 86, 1470, 913, 1774, 948, 1509, 1022 1756,130, then, derive the total head movements to satisfy all the pending requests for FCFS, SSTF, LOOK and SCAN disk scheduling algorithms..

Solution:

1. Solution:

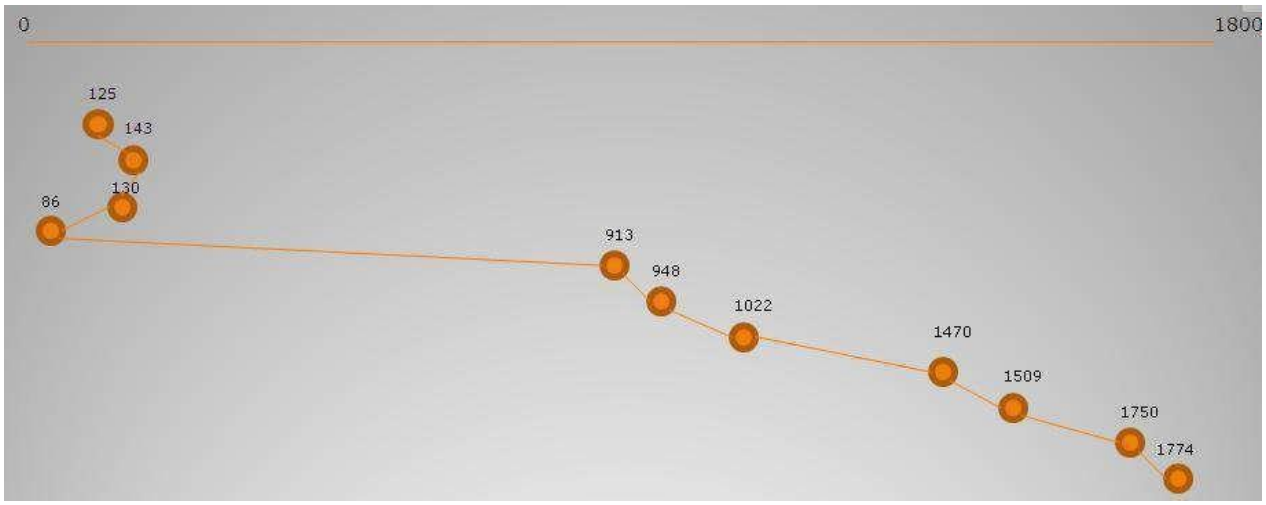
(i) FCFS



From cylinder	To cylinder	Seek Time
143	86	57
86	1470	1384
1470	913	557
913	1774	861
1774	948	826
948	1509	561
1509	1022	487
1022	1750	728
1750	130	1620
Total Seek Time		7081

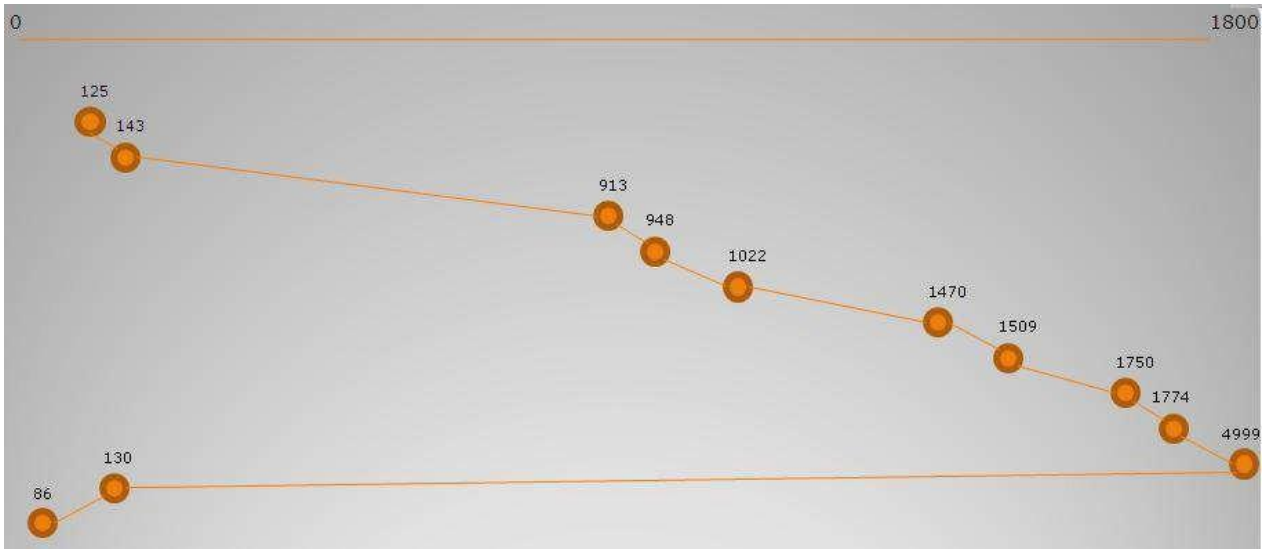
For FCFS schedule, the total seek distance is 7081.

(ii) SSTF



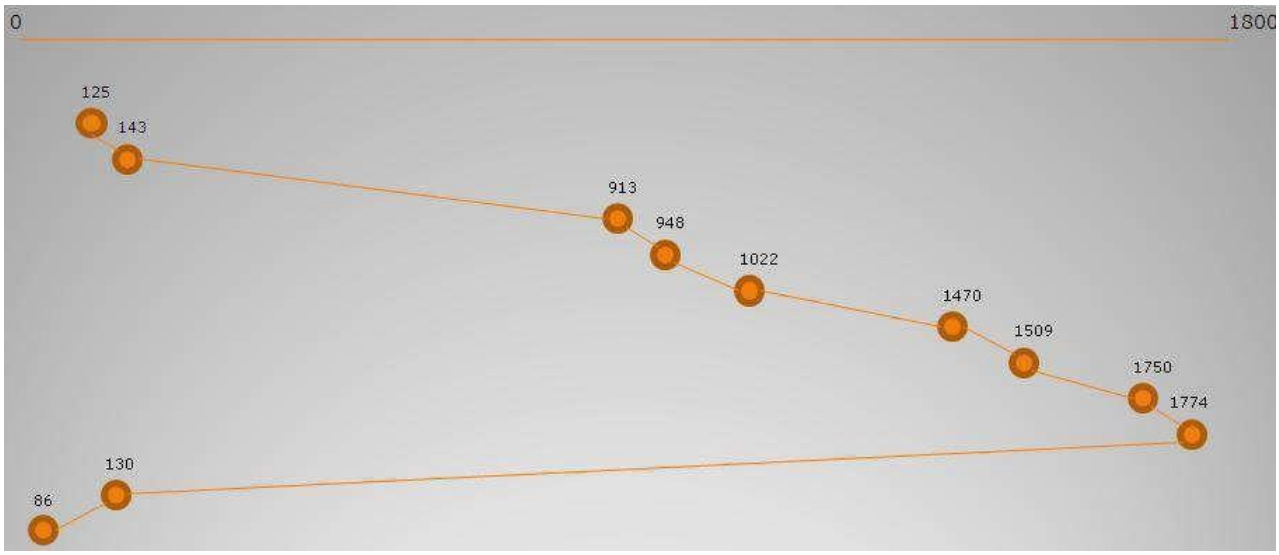
For SSTF schedule, the total seek distance is 1745.

(iii) SCAN



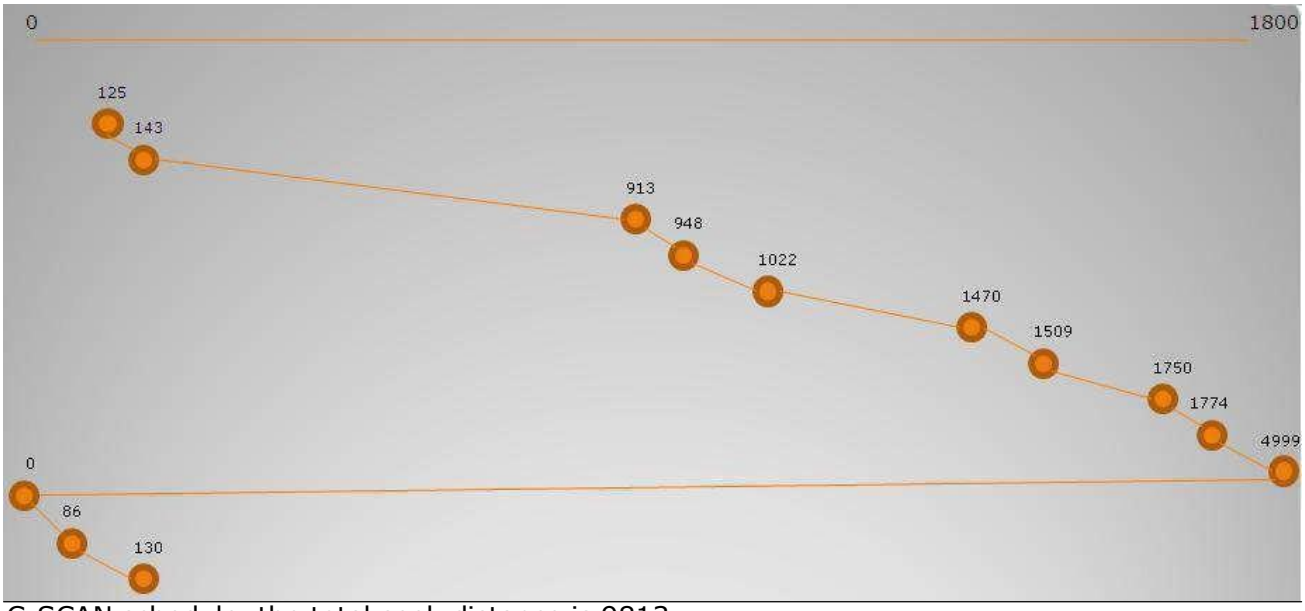
For SCAN schedule, the total seek distance is 9769.

(iv) LOCK



For LOOK schedule, the total seek distance is 3319.

(v) C-SCAN



For C-SCAN schedule, the total seek distance is 9813.

4. What is access matrix? Explain the different methods of implementing access matrix.

Solution:

- Access-matrix provides mechanism for specifying a variety of policies.
- The access matrix is used to implement policy decisions concerning protection.
- In the matrix, 1) Rows represent domains.
  2) Columns represent objects.
  3) Each entry consists of a set of access-rights (such as read, write or execute).
- In general, Access(i, j) is the set of operations that a process executing in $Domain_i$ can invoke on $Object_j$
- Example: Consider the access matrix shown in Figure 5.10.
  - There are
    1) Four domains: D_1, D_2, D_3, and D_4
    2) Three objects: F_1, F_2 and F_3
  - A process executing in domain D_1 can read files F_1 and F_3.

domain \ object	F_1	F_2	F_3
D_1	read		read
D_2			
D_3		read	execute
D_4	read write		read write

Figure 5.10 Access matrix

- Domain switching allows the process to switch from one domain to another.
- When we switch a process from one domain to another, we are executing an operation (switch) on an object (the domain)
- We can include domains in the matrix to control domain switching.
- Consider the access matrix shown in Figure 5.11.
  - A process executing in domain D_2 can switch to domain D_3 or to domain D_4.

domain \ object	F_1	F_2	F_3	D_1	D_2	D_3	D_4
D_1	read		read		switch		
D_2						switch	switch
D_3		read	execute				
D_4	read write		read write	switch			

Figure 5.11 Access matrix with domains as objects

- Allowing controlled change in the contents of the access-matrix entries requires 3 additional operations (Figure 5.12):
  1) **Copy(*)** denotes ability for one domain to copy the access right to another domain.
  2) **Owner** denotes the process executing in that domain can add/delete rights in that column.
  3) **Control** in access(D_2,D_4) means: A process executing in domain D_2 can modify row D_4.

domain \ object	F_1	F_2	F_3	D_1	D_2	D_3	D_4
D_1	read		read*		switch		
D_2						switch	switch control
D_3		read owner	execute				
D_4	write		write	switch			

Figure 5.12 Access matrix with Copy rights, Owner rights & Control rights

- The problem of guaranteeing that no information initially held in an object can migrate outside of its execution environments is called the confinement problem.

Implementation of Access Matrix

Global Table

- A global table consists of a set of ordered triples <domain, object, rights-set>.
- Here is how it works:
  - Whenever an operation M is executed on an object O_j within domain D_i, the global table is searched for a triple $< D_i , O_j , R_k >$, with $M \in R_k$.
  - If this triple is found,
    Then, we allow the access operation;
    Otherwise, access is denied, and an exception condition occurs.
- Disadvantages:
  1) The table is usually large and can't be kept in main memory.

2) It is difficult to take advantage of groupings, e.g. if all may read an object, there must be an entry in each domain.

Access Lists for Objects

- In the access-matrix, each column can be implemented as an access-list for one object.
- Obviously, the empty entries can be discarded.
- For each object, the access-list consists of ordered pairs <domain, rights-set>.
- Here is how it works:
  - Whenever an operation M is executed on an object O_j within domain D_i, the access-list is searched for an entry $< D_i , R_k >$, with $M \in R_k$.
  - If this entry is found,

    Then, we allow the access operation; Otherwise, we check the default-set.

    If M is in the default-set, we allow the access operation; Otherwise, access is denied, and an exception condition occurs.
- Advantages:

1) The strength is the control that comes from storing the access privileges along with each object
2) This allows the object to revoke or expand the access privileges in a localized manner.

- Disadvantages:

1) The weakness is the overhead of checking whether the requesting domain appears on the access list.
2) This check would be expensive and needs to be performed every time the object is accessed.
3) Usually, the table is large & thus cannot be kept in main memory, so additional I/O is needed.
4) It is difficult to take advantage of special groupings of objects or domains.

Capability Lists for Domains

- For a domain, a capability list is a list of objects & operations allowed on the objects.
- Often, an object is represented by its physical name or address, called a capability.
- To execute operation M on object O_j, the process executes the operation M, specifying the capability (or pointer) for object O_j as a parameter.
- The capability list is associated with a domain.

  But capability list is never directly accessible by a process.

  Rather, the capability list is maintained by the OS & accessed by the user only indirectly.
- Capabilities are distinguished from other data in two ways:

3) Each object has a tag to denote whether it is a capability or accessible data.
4) Program address space can be split into 2 parts.
   i) One part contains normal data, accessible to the program.
   ii) Another part contains the capability list, accessible only to the OS

A Lock–Key Mechanism

- The lock–key scheme is a compromise between 1) Access-lists and 2) Capability lists.
- Each object has a list of unique bit patterns, called locks.
- Similarly, each domain has a list of unique bit patterns, called keys.
- A process executing in a domain can access an object only if that domain has a key that matches one of the locks of the object.

5. Explain the different types of directory structures with examples and mention their advantages and disadvantages.

Solution:

- Operations performed on a directory:

1) Search for a File

- We need to be able to search a directory-structure to find the entry for a particular file.

2) Create a File

- We need to be able to create and add new files to the directory.

3) Delete a File

- When a file is no longer needed, we want to be able to remove it from the directory.

4) List a Directory

- We need to be able to
  → list the files in a directory and
  → list the contents of the directory-entry for each file.

5) Rename a File

- Because the name of a file represents its contents to its users, we must be able to change the name when the contents or use of the file changes.

6) Traverse the File-system

- We may wish to access
  → every directory and
  → every file within a directory-structure.
- For reliability, it is a good idea to save the contents and structure of the entire file-system at regular intervals.

Single Level Directory

- All files are contained in the same directory (Figure 4.19).
- Disadvantages (Limitations):
  7) Naming problem: All files must have unique names.
  8) Grouping problem: Difficult to remember names of all files, as number of files increases.

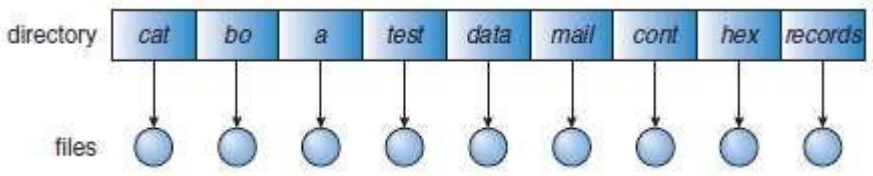


Figure 4.19 Single-level directory

Two Level Directory

- A separate directory for each user.
- Each user has his own UFD (user file directory).
- The UFDs have similar structures.
- Each UFD lists only the files of a single user.
- When a user job starts, the system's MFD is searched (MFD=master file directory).
- The MFD is indexed by user-name.
- Each entry in MFD points to the UFD for that user (Figure 4.20).

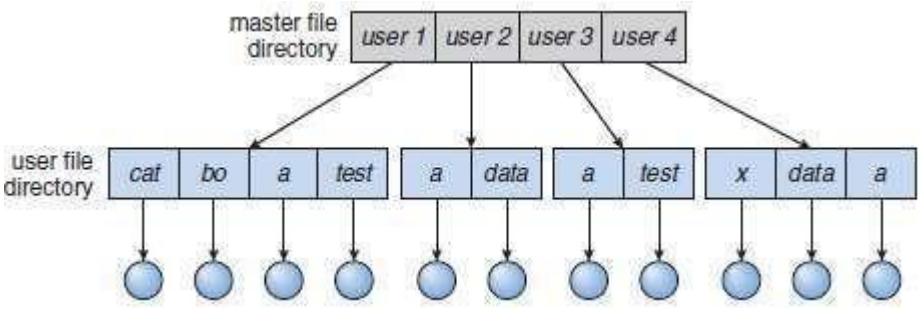


Figure 4.20 Two-level directory-structure

- To create a file for a user,
  the OS searches only that user's UFD to determine whether another file of that name exists.
- To delete a file,

the OS limits its search to the local UFD. (Thus, it cannot accidentally delete another user's file that has the same name).

- Advantages:
  9) No filename-collision among different users.
  10) Efficient searching.
- Disadvantage:
  1) Users are isolated from one another and can‟t cooperate on the same task.

Tree Structured Directories

- Users can create their own subdirectories and organize files (Figure 4.21).
- A tree is the most common directory-structure.
- The tree has a root directory.
- Every file in the system has a unique path-name.

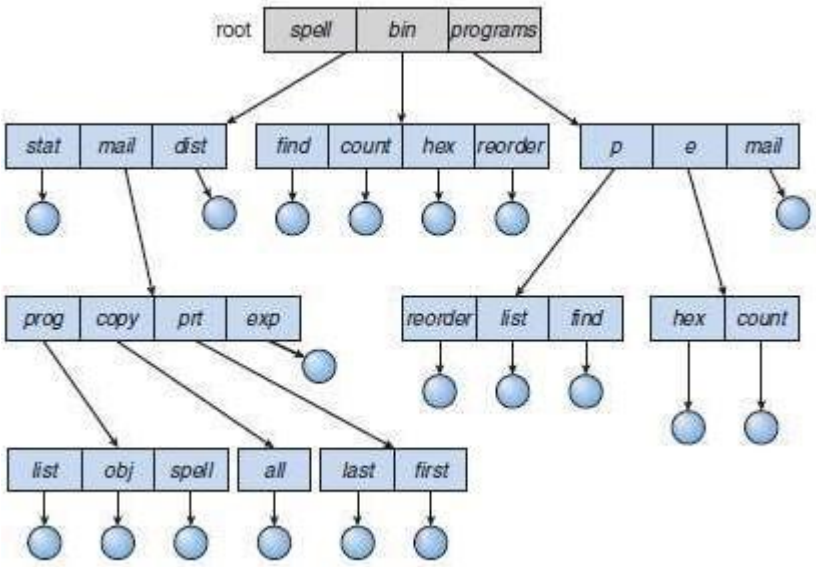


Figure 4.21 Tree-structured directory-structure

- A directory contains a set of files (or subdirectories).
- A directory is simply another file, but it is treated in a special way.
- In each directory-entry, one bit defines
  as file (0) or
  subdirectory (1).
- Path-names can be of 2 types:
- Two types of path-names:
  11) **Absolute path-name** begins at the root.
  12) **Relative path-name** defines a path from the current directory.
- How to delete directory?
  1) To delete an **empty directory**:
     → Just delete the directory.
  2) To delete a **non-empty directory**:
     → First, delete all files in the directory.
     → If any subdirectories exist, this procedure must be applied recursively to them.
- Advantage:
  1) Users can be allowed to access the files of other users.
- Disadvantages:
  1) A path to a file can be longer than a path in a two-level directory.
  2) Prohibits the sharing of files (or directories).

Acyclic Graph Directories

- The directories can share subdirectories and files (Fig(An **acyclic graph** means a graph with no cycles).
- The same file (or subdirectory) may be in 2 different directories.
- Only one shared-file exists, so any changes made by one person are immediately visible to the other.

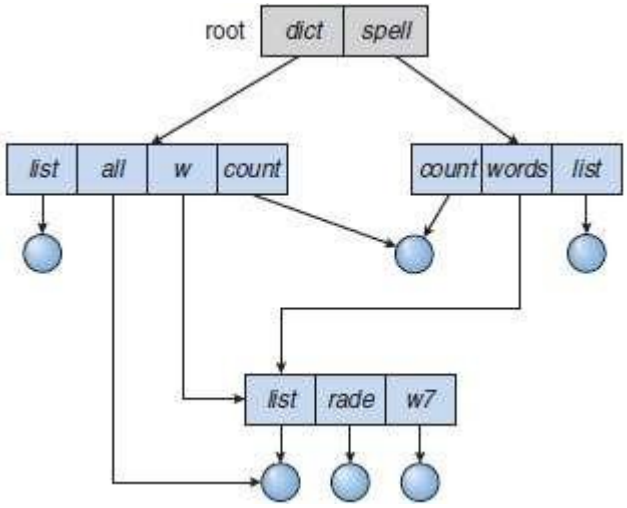


Figure 4.22 Acyclic-graph directory-structure

- Two methods to implement shared-files(or subdirectories):
  13) Create a new directory-entry called a link.
      A link is a pointer to another file (or subdirectory).
  14) Duplicate all info. about shared-files in both sharing directories.
- Two problems:
  1) A file may have multiple absolute path-names.
  2) Deletion may leave dangling-pointers to the non-existent file. Solution to deletion problem:
  1) Use backpointers: Preserve the file until all references to it are deleted.
  2) With symbolic links, remove only the link, not the file. If the file itself is deleted, the link can be removed.

General Graph Directory

- Problem: If there are cycles, we want to avoid searching components twice (Figure 4.23).
  Solution: Limit the no. of directories accessed in a search.
- Problem: With cycles, the reference-count may be non-zero even when it is no longer possible to refer to a directory (or file). (A value of 0 in the reference count means that there are no more references to the file or directory, and the file can be deleted).
  Solution: Garbage-collection scheme can be used to determine when the last reference has been deleted.
- Garbage collection involves
  15) First pass
      → traverses the entire file-system and
      → marks everything that can be accessed.
  16) A second pass collects everything that is not marked onto a list of free-space.

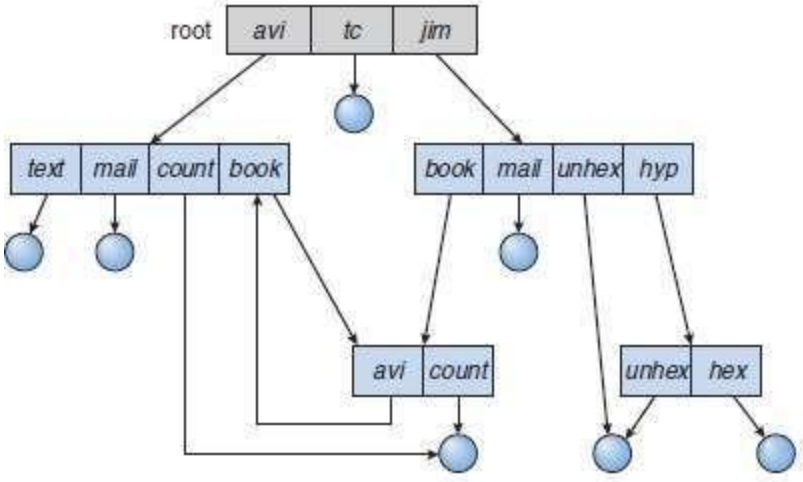


Figure 4.23 General graph directory

6. Explain the performance of demand paging

Solution:

Paging

- A demand-paging system is similar to a paging-system with swapping (Figure 4.2).
- Processes reside in secondary-memory (usually a disk).
- When we want to execute a process, we swap it into memory.
- Instead of swapping in a whole process, **lazy swapper** brings only those necessary pages into memory.

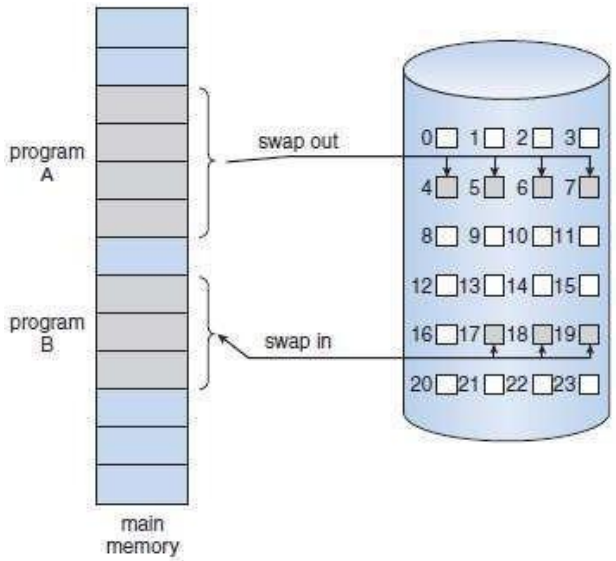


Figure 4.2 Transfer of a paged memory to contiguous disk space

Basic Concepts

- Instead of swapping in a whole process, the pager brings only those necessary pages into memory.
- Advantages:
  1) Avoids reading into memory-pages that will not be used,
  2) Decreases the swap-time and
  3) Decreases the amount of physical-memory needed.
- The valid-invalid bit scheme can be used to distinguish between
  → pages that are in memory and
  → pages that are on the disk.
  1) If the bit is set to **valid**, the associated page is both legal and in memory.

2) If the bit is set to **invalid**, the page either
→ is not valid (i.e. not in the logical-address space of the process) or
→ is valid but is currently on the disk (Figure 4.3).

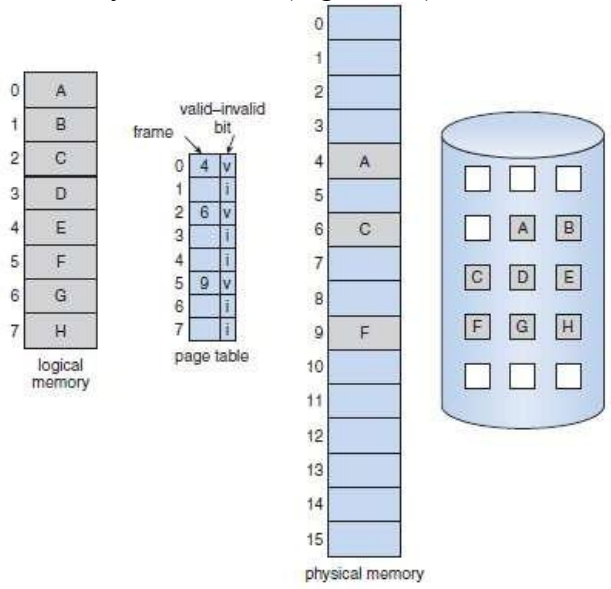


Figure 4.3 Page-table when some pages are not in main-memory

- A **page-fault** occurs when the process tries to access a page that was not brought into memory**.**
- Procedure for handling the page-fault (Figure 4.4):

7) Check an internal-table to determine whether the reference was a valid or an invalid memory access.
8) If the reference is invalid, we terminate the process.
If reference is valid, but we have not yet brought in that page, we now page it in.
9) Find a free-frame (by taking one from the free-frame list, for example).
10) Read the desired page into the newly allocated frame.
11) Modify the internal-table and the page-table to indicate that the page is now in memory.
12) Restart the instruction that was interrupted by the trap.

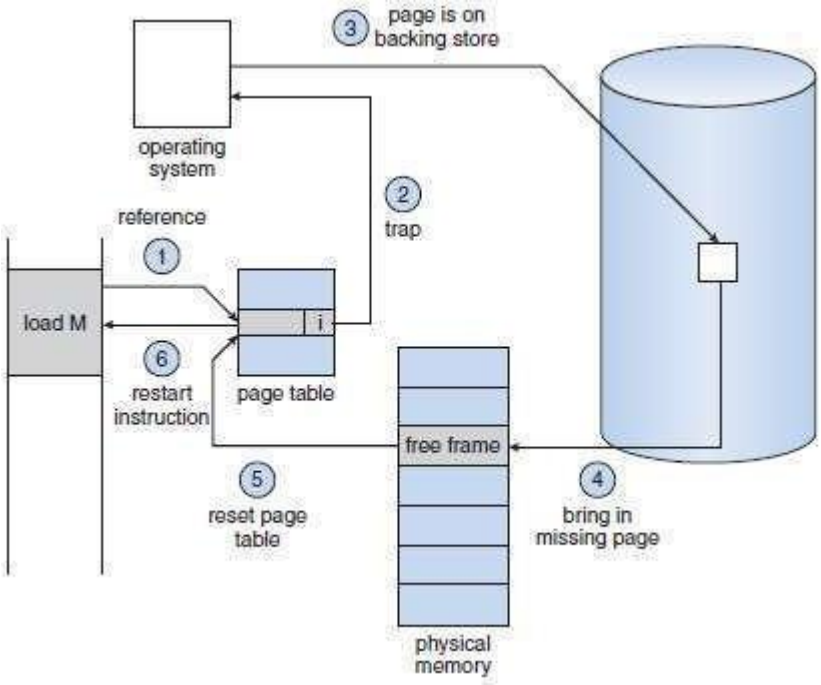


Figure 4.4 Steps in handling a page-fault

- **Pure demand paging:** Never bring pages into memory until required.
- Some programs may access several new pages of memory with each instruction, causing multiple page-faults and poor performance.
- Programs tend to have locality of reference, so this results in reasonable performance from demand paging.
- Hardware support:

1) Page-table

- Mark an entry invalid through a valid-invalid bit.

2) Secondary memory

- It holds pages that are not present in main-memory.
- It is usually a high-speed disk.
- It is known as the **swap device** (and the section of disk used for this purpose is known as swap space).

Performance

- Demand paging can significantly affect the performance of a computer-system.
- Let p be the probability of a page-fault ($0 \leq p \leq 1$). if p = 0, no page-faults.
  if p = 1, every reference is a fault.
- effective access time(EAT)=[(1 - p) *memory access]+ [p *page-fault time]
- A page-fault causes the following events to occur:

4) Trap to the OS.
5) Save the user-registers and process-state.
6) Determine that the interrupt was a page-fault. '
7) Check that the page-reference was legal and determine the location of the page on the disk.
8) Issue a read from the disk to a free frame:
   a. Wait in a queue for this device until the read request is serviced.
   b. Wait for the device seek time.
   c. Begin the transfer of the page to a free frame.
9) While waiting, allocate the CPU to some other user.
10) Receive an interrupt from the disk I/O subsystem (I/O completed).
11) Save the registers and process-state for the other user (if step 6 is executed).
12) Determine that the interrupt was from the disk.
13) Correct the page-table and other tables to show that the desired page is now in memory.
14) Wait for the CPU to be allocated to this process again.
15) Restore the user-registers, process-state, and new page-table, and then resume the interrupted instruction.