ASSIGNMENT

Submitted By:

Muhammad Bilal

Roll no:

Mtf24003509

Subject:

Introduction To Operating System

Submitted to:

Prof. Zafar Kamrani

Q1: Explain how the operating system handles memory allocation and deallocation for processes, and describe the key techniques (e.g., paging, segmentation) used to manage memory efficiently in a multi-tasking environment.

Answer:

In Operating Systems, Memory Management. The operation of the OS for the function of memory management is right on the verge. This will ensure a smooth and a safe utilization of the memory present in the computer. It will allocate to the processes memory required and free from it when it is no longer needed. It will also provide techniques for making memory utilization more efficient, especially during a multitasking environment.

Here is how the OS handles the allocation of memory: Static Allocation:

Definition: Memory allocation during compile time is to a process and is, therefore, unaltered throughout the program execution. Example: Global and static variables in languages like C/C++. Limitation: It's somewhat inflexible, for there's no option for dynamic reallocation of memory according to runtime needs. Dynamic Allocation:

Definition: Memory allocation which occurs when the program runs. Examples: - Heap allocation. Use of functions like malloc() in C or new in C++. - Stack allocation. Memory is allocated for local variables/live data with the runtime generation of function call frames. By doing so, the program benefits from an effective memory allocation: only needed memory will be allocated. Deallocation:

Automatic Deallocation: When the process terminates, frees its memory by the OS. Manual Deallocation: Programs free memory using system calls (for example, using free() in C). Garbage Collection: Automatically frees the allocated memory in high-level languages like Python or Java, allowing for unused memory.

Key Techniques for Efficient Memory Management

Memory management techniques are to ensure the optimal use of available memory while keeping health and performance of the system sound. Here is a detailed description of the mentioned techniques:

Paging

How It Works:

Memory Division: The physical memory is divided into fixed size blocks called frames and the logical memory (for use of the processes) into similar blocks called pages.

Mapping: Thus, the operating system maps the pages into the frames, thereby enabling non-contiguous occupancy of the pages in the physical memory by a process.

Translation: Translation of logical to physical address happen by the CPU using a page table during execution.

Advantages:

No External Fragmentation:

With fixed-size pages there are no holes in memory that may occur with variable-sized memory allocation.

Memory is used by a more efficient manner since the space does not need to be found for large contiguous blocks for processes.

Allows Multitasking:

It would make possible the co-existing of pages belonging to different processes in physical memory at the same time thus enabling concurrent execution.

Disadvantages:

Internal Fragmentation:

The page may not be entirely filled with data and thus there is some leftover space (that may happen when the process does not fill the whole memory allocated).

Wasted memory is created by these unutilized parts.

Overhead:

The page table incurs added computational and memory costs for managing.

It adds latency in the logical to the physical address translation.

Segmentation:

How It Works:

Segmentation indicates logical division where segments may be of different sizes. Each segment corresponds to some logical components of a program like code data stack.

Allocation: Each segment gets a continuous area in physical memory and a segment table which logically maps segments to their location in terms of physical addresses.

Advantages:

Logical Program Structure:

Segments fit the logical structure of a program and prove to make easy debugging and development for a program.

Logical divisions are easy to visualize and manage.

Memory Protection:

Isolate the types of memory such as code, stack, which increase reliability and security.

Disadvantages:

External fragmentation:

Unlike homogenous-sized segments, where gaps are formed in memory on deallocation, space gets wasted due to inefficient memory usage.

Complex Management:

Allocation in variable-sized memory segments is a very complicated process. File fragmentation may be needed to free a section of memory.

Segmentation:

How It Works:

Memory parts are divided into segments of different sizes that form logical code, data, stack, and so on.

The allocation of them is made such that each segment is allocated a contiguous block in physical memory, while a segment table maps logical segments to physical addresses.

Advantages:

Logical Program Structure:

All segments correspond to a logical structure of the program that will facilitate debugging of programs along with development.

Logical divisions will be easy to visualize and manage.

Memory Protection:

The different types of memory like code and stack will be isolated and will increase reliability and even security.

Disadvantages:

External Fragmentation: segments have different sizes owing to which, when deallocated, space will be wasted due to ineffective usage of memory in terms of wasted gaps.

Complex Management: Allocation and deallocation are a very complex process by variable-sized memory segments; and re-fragmentation is a necessity to free a section of memory.

Q2:Discuss the concept of swapping in operating systems. How does it work, and what is its impact on system performance, particularly in situations of low available physical memory?

Moving inactive or unused processes/pages to secondary storage and swapping them back when necessary.

Benefits:

Support For Large Programs:

This enables the execution of programs that would otherwise not fit into available RAM. It does so by loading data into memory only as it is needed.

Process Isolation:

Isolation of Processes.

Each process now is running in its own virtual address space. So no processes can interfere with each other.

Multitasking

Gets multiple processes to run at the same time and swaps those which are inactive to storage secondary.

Memory Protection and Isolation

How It Works :

Hardware Mechanisms :

Base and Limit Registers : Enforce processes operate solely with their allowable memory.

Virtual Memory Mapping : For process-specific virtual addresses into a specific address space, into an isolation condition

Access Control:

No Process can access or change other processes' memory unless explicitly given permission (e.g., shared memory).

Importance :

That corrupts data : By isolating memory, one process cannot overwrite others' memory space and therefore keep its integrity intact.

Increases Security:

Prevents malicious processes from accessing sensitive data in another process's memory space.

Facilitates Multiple Users Systems:

Allows multiple users to run safely their processes in the same system without interference.

Q3: In contiguous memory allocation, a process is assigned a contiguous block of memory. What are the key advantages and drawbacks of using this allocation method, and how does external fragmentation affect it?

Ans: In contiguous memory allocation, a single continuous portion area of memory is assigned to each process. This scheme is straightforward and efficient for accessing memory, but it also has some merits and demerits.

Some Advantages:

Ease of implementation: It is fairly simple to manage and implement because each process is assigned just one memory block.

The quick access: Since block are contiguous, they can quickly access memory without any elaborate address calculation or pointers.

No intricate memory management is required: No advanced memory management schemes are required, such as paging or segmentation. Therefore, this configuration can be used even in resource-limited systems.

Major Disadvantages:

Lesser Flexibility: Suppose a process needs more memory than that which was allocated to it; it is difficult to enlarge its memory block because there may not be any other free memory contiguous to it. This would thus lead to poor utilization of space.

Memory Wastage: For very small processes, a huge block of contiguous memory would simply go to waste, resulting in an inefficient memory utilization.

External Fragmentation: The memory will get fragmented into smaller pieces when objects get loaded into and out of the system. After some time, the system would be in a situation where it might have enough total free space but nothing contiguous that would be large enough to hold a particular process. This is termed as External Fragmentation. This can, over time, cause very adverse effects on the efficiency of memory allocation on the system.

So this fragmentation, in turn, can affect memory usages and it may lead to errors during memory allocations. Usually, errors occur because there is not enough contiguous space available for the procedure, though, overall, enough memory may exist in the system.

Fragmentation Solutions:

Compaction: Moving processes from time to time will create larger contiguous blocks of free memory by the OS, though a significant performance penalty is incurred.

Segmentation or paging: These can be considered as substitutes for contiguous memory allocation without bringing problems of fragmentation.

Q4: Compare and contrast the concepts of segmentation and paging in memory management. How does each method handle memory allocation, and what are the benefits and limitations of each approach?

Segmentation:

Concept: Segmentation is a memory management scheme in which processes are traced into logical segments such as code, data, stack etc. Each segment is a separate entity and can be placed anywhere in the physical memory.

Memory Assignment:

Segments may vary in size according to the logical structure of the program (for example, a code segment may be larger than a stack segment).

Each segment gets a contiguous piece of memory; however, all segments may not be contiguous from the view of different processes.

Advantages:

Logical Organization: Segmentation reflects the logical structure of the program which simplifies understanding and management.

Handling different types of data more efficiently: Different segments can have different lengths, thus providing better memory usage for some types of data (ex. stack segments can be increased or decreased according to need).

Downfalls:

External fragmentation: Because of processes being loaded and removed, free memory is scattered into smaller topologies, which leads to external fragmentation and makes it harder and more difficult to give large consecutive blocks of memory.

The company engages in the management of Information: Encompassed in the above condition, information is also to be managed in terms of several segments in a process which increases complexity in memory managing systems.Paging:

Concept: The physical memory is divided into blocks of fixed sizes which are referred to as pages and logical memory or virtual memory into blocks of fixed size referred to as page frame. The page can be mapped to any available page frame in physical memory.

Memory Allocation:

Pages are of equal size and processes are divided into pages. Those pages are mapped to non-contiguous frames in physical memory.

Paging keeps away from the need of contiguous memory allocation which avoids fragmentation issues typically found in segmentation.

Advantages:

There is absolutely no external fragmentation; so, pages can be placed at any point in the physical memory. Therefore, there is no external fragmentation at all; hence, memory can also be used very well.

Simple management of memory. With fixed-size pages, tracking memory allocation becomes easier.

The paging system supports the concept of virtual memory by which a process can avail a much larger amount of memory than it has physically.

Disadvantages:

Internal fragmentation: Memory is wasted in the page wherein everything is not generally used (partially used by a process), wastage of part of a page.

Overhead due to page table management: Since the operating system manages the page table for virtual memory to physical memory mapping, additional space has to be allocated.

Q5: What is virtual memory, and how does it allow processes to exceed the limits of physical memory? Discuss how the operating system uses paging or segmentation to implement virtual memory, including the role of the page table and page faults.

What is Virtual Memory?

Virtual memory is a method of memory management that makes use of the hardware and the software of the operating system in order to compensate for shortages in the amount of physical memory available in the computer. The memory gave users a false impression that they had a very large block of contiguous memory even when the physical memory was much smaller.

Virtual memory gives processes a view into physical memory that is larger than the actual amount of RAM by swapping memory content between physical and secondary storage.

How Virtual Memory Allows Surpassing of Hardware Memory Limits:

Swapping: Virtual memory allows processes to exceed physical memory capabilities with swapping of segments or pages between RAM and disk storage. When a process needs a page that is not in physical memory, the operating system retrieves it from disk (a page fault occurs).

Operating systems manage the parts of processes kept in memory and those swapped out to disk using algorithms like least recently used (LRU) or first-in-first-out (FIFO). Page Table Mapping of Virtual Memory to Physical Memory:

Paging: In paging system, virtual memory divided into fixed size blocks called pages; in physical memory, pages correspond with equal size fixed blocks called page frames. Operating system maintains a page table for mapping virtual pages with the physical ones.

In the case of an access by a process to a page that is not currently in physical memory (a page fault), the operating system loads the page from disk into an available frame, swapping out another page if it has to.

Page Table:

A page table is a structure that stores the mapping between virtual pages and physical page frames. It contains an entry against each page which provides the physical memory address of the corresponding page or indicates that the page is on disk.

It allows the operating system to quickly convert virtual addresses into physical addresses.

Page Faults:

This happens when a process tries to access a page that is not in physical memory. The operating system loads that page into memory and, if this results in a full memory, will swap out one of the pages to make space.

Segmentation:

Virtual memory, in a segmentation system, divides a virtual address space into segments, such as code, data, stack (the actual kinds of segments defined in the architecture might vary). It may create a number of non-contiguous physical memory areas for a segment. The operating system loads that segment into memory from the disk when not resident in memory.

The segment table, in that case, is used for mapping virtual segments to physical memory addresses. When a segment is not in memory, it leads to a segment fault, and the OS loads the segment from disk.

Summary of Virtual Memory Management:

This technique enables processes to see an abstract view of memory that is much more than what is actually available on physical memory. Paging or segmentation are virtual memory management techniques.

Paging breaks both virtual and physical memories into fixed-size blocks and maps them through a page table.

Segmentation creates logical divisions in memory (segments, in our case), maps to physical memory, and the sizes are more flexible compared to paging.

The operating system handles page fault or segment faults by swapping pages or segments in or out of memory as needed.

Q6: Describe the concept of demand paging and explain how the operating system handles page faults when a page that is needed by a process is not currently in memory. How does the page replacement algorithm come into play during this process?

Demand Paging

By demand paging, we mean demand paging is a type of memory management by which pages of the process are loaded into memory only when they are needed, that is, when the process accesses them. Rather than having the entire process loaded into memory all at once, this will conserve memory.

Handling Page Faults:

When a page fault occurs, a process is trying to access a page that it does not currently have in memory.

An overview on how the operating system acts when a page fault situation arises is as follows:

It checks whether space is available in memory to load the missing page.

When there is no sufficient space in memory, then it chooses a page to swap out (using any page replacement policy).

Then the required page is read from the disk and loaded into main memory.

The page table is updated to reflect the current location of the page in memory, and the process continues.

Page Replacement Algorithms:

LRU (Least Recently Used): Removes the unstored page that is not used for the longest period of time.

FIFO (First-In, First-Out): The oldest page replaces the oldest page regardless of how recently it was used.

Optimal: Replace the page that will be used most distant in the future (ideal, but difficult to implement as a practical rule).

These algorithms help from the point of view of the OS regarding which page to swap out in a non-free state.

Q:7 What is thrashing in the context of memory management, and how does it occur when the system is under heavy load? Discuss the conditions that lead to thrashing and how an operating system can prevent or mitigate it.

What is Thrashing?

Thrashing occurs when an operating system spends more time swapping pages than executing processes because physical memory isn't big enough to hold the current number of running processes.

How Thrashing Occurs:

It happens when the number of processes becomes very large in the system and each process needs more and more page access than available. Thus, all the required pages are not present in memory and the operating system must spend a lot of time swapping pages in and out rather than running the processes.

This leads to a severe dip in the performance of the system.

Conditions for Thrashing:

Overcrowding of processes: Because of the excessive number of active processes, the memory required by each is higher than what is available.

Underutilized memory resources: Not enough physical memory is available to the system, which results in a kind of finish exhaustion of memory, leading to frequent pages faults and swapping.

Under-efficient page replacement algorithms: Inefficient algorithms can cause extensive swapping leading to wastage in memory use.

Ways to Avoid or Cure Thrashing:

Reducing the active processes: The OS can suspend processes altogether for time usage or in some ways limit the number of processes that are kept active to avoid memory overloading.

Page fault frequency (PFF): It can measure the number of page faults and then reduce the number of processes executing if the page fault count goes above a threshold.

Better page replacement algorithms: Using algorithms like LRU or Optimal can help to cut off unnecessary page movements.

Increasing physical memory: More RAM added to the system can minimize thrashing by offering more space to the processes.

By optimizing resource use and effective memory management, it prevents thrashing and can improve system performance at the same time.

Q8: Explain the concept of memory-mapped files. How do memory-mapped files differ from traditional file I/O operations, and what are the performance benefits of using memory-mapped files for large data sets?

This is a mechanism that allows applications to treat the files as if they are directly in memory, as part of the process virtual memory, giving them the advantage of quicker data access because the file is merely mapped into the process memory address space, with changes made on the memory updated in the original file.

Differences from Traditional File I/O:

In traditional file I/O, to read/write data from/to a file, a program uses the system calls like read() or write(). Each operation triggers a context switch from the program to the operating system and back, increasing overhead.

Memory-Mapped Files: The contents of the file are mapped by the OS into the address space of the process with memory mapping. Applications may now use the data as if it were indeed a part of its memory. No explicit read and write operations need doing, and the I/O handling is done in the background by the whole system.

Performance Benefits for Large Data Sets:

Faster Access: Memory-mapped files make it much faster to cross memory boundaries. They provide access to a real part of the file without annoying disk I/O overhead. This way, one does not have to go back and forth between the user/kernel spaces.

Memory Usage Efficiency: Memory usage efficiency is enhanced in that the operating system can load only into memory those parts of the file needed at a given moment (demand paging).

Shared Memory: Memory-mapped files can be shared among many processes. For this reason, they are particularly interesting in inter-process communication (IPC) because they allow different processes to access the same data set without physically duplicating it in memory.

Q:9 What are the primary components of a file system, and how does an operating system manage file storage, access, and retrieval? Discuss how different file systems (e.g., FAT, NTFS, ext4) handle file organization and metadata.

Typically, where the disk is used as a storage medium, a file system is needed to manage the stored files. Primary Components of a File System:

File Control Block (FCB): The structure of data that maintains metadata related to each file, such as the name of the file, size, location on the disk, access permissions, etc.

Directory Structure: This organizes files into directories (or folders) to make it easier for users and applications to find the file.

File Data: This is actual content of the files stored on disk.

Free Space Management: A process that records unused disk space and determines the location where new data could be written.

How Operating System Controls File Storage, Access, and Retrieval:

Storage: Storing files in openspace could vary from the file systems structure allocated by the operating system on the disk. It then keeps a record of where these files are located using file allocation tables, pointers to data blocks, etc.

Access: The OS would retrieve the requested data either directly on the disk through reading or employing some memory-mapping techniques.

Retrieval: It uses an index or linkage (e.g., file allocation tables, inode tables) to determine the location in the disk where the file is found, and it maps this into the memory for access.

Different File Systems (FAT, NTFS, ext4) and How They Organize Files and Manage Metadata- FAT (File Allocation Table)-

Organization: FAT is a very simple table that keeps track of the location of clusters on the disk where the file is stored. Each entry of that table then corresponds to that cluster and the next cluster number it will have in that file.

Metadata: It also includes basic metadata for the file, such as file name, creation date, file size, etc.; however, it cannot handle complex file attributes very well. Shortcomings: It merely does not have support for advancements like file permissions, journaling, and, moreover, it is not efficient to work well in case of large volumes or big file sizes.

NTFS (New Technology File System)-Organization: The Master File Table is a table that contains metadata for the storage of all files and directories in NTFS. MFT information like file name, timestamp, security attributes, and data block locations are present in MFT. Metadata: The rich metadata that includes permissions, security descriptors with an extensive range of file properties such as file compression, as well as journaling, is the model by which NTFS works.

Benefits: The efficiencies of NTFS include large file sizes, advanced features such as encryption, compression of files, and access control making it optimal for present-day systems.

ext4 (Fourth Extended File System):

Organization: Inodes are used by ext4 to organize data. These link together file metadata including size, permissions, and modification times. Associated with inodes are data blocks where actual file data is stored. Metadata: ext4 provides such modern features for file systems as journaling, which is also available on NTFS, extended file attributes, and permits larger file sizes and volumes than older systems such as ext3. Strengths: ext4 has very high performance and reliability, especially with large files, and it includes features such as delayed allocation and checksumming.

Q:10 What is the concept of a file in an operating system? Explain the different types of files (e.g., regular files, directories, special files) and how the operating system abstracts and manages file access for users and applications.

A file in an operating system is merely a facility that provides a way for users to store a collection of data or information on a storage device like a hard disk or an SSD. It represents almost everything known in the outside world, such as a text file, program, image, or even configuration setting. In short, files are considered as units of storage in any computer system at the base level.

The operating system (OS) abstracts the files for easy access to users or applications. This provides the use of opening, reading, writing, or closing files without requiring any knowledge about how the files are physically structured under the hood.

Types of Files:

Regular files: These are the most common files. They hold specific data for the end-user, such as text files, photographs, videos, or programs; they serve the functions of a commons file of a file system with a name, permissions, and some metadata associated therein.

Directories: Directories are a special kind of file that contains a list of other files and directories. A directory holds the files in a hlperarchy and makes it possible for the user to get access to the files more easily.

A directory file holds the names and locations of other files (or subdirectories) to organize data in a tree-like manner.

Special Files: Special files are abstractions of devices or interfaces through which programs access to interact with some hardware devices or system resources is provided. These include:

Character special files (e.g., terminal devices) for communication on a byte-by-byte basis.

Block special files (e.g., hard drives or storage devices) for accessing blocks of data in fixed-size chunks.

Socket files for inter-process communication (IPC).

Q:11 How are directories organized within a file system, and what role do they play in managing files? Explain the structure of a disk and the relationship between sectors, clusters, and tracks in terms of data storage and retrieval.

Structure of Directories and Disks

How Directories Are Organized Within a File System:

Directories are places to organize and govern files inside a file system. Dirctories organize files logically so that users can find their data easily and manage it without difficulty.

Directories create a hierarchical tree structure. The top level is usually the root directory, which has files under it or subdirectories. Within each of these are more subdirectories and/or files. Thus the tree of files and directories is created.

In some systems, directories also can have some special information concerning an access control list (ACL) or timestamps, or even file attributes for files contained by it.

The operating system keeps a directory table for each directory it maintains in the system, which contains the name and pointer (or address) to files and other directories in that directory structure.

Role of Directories:

Organizing Files: Directories logically organize files, hence enabling both users and applications to locate and access them easily.

Access Control. Directories also help manage access controls over files that reside in them, taking care of who can read, write or execute.

Pathnames Include: Directories also provide a relative or absolute pathname to help traverse across the filesystem in reference to file locations.

Structure of a Disk:

A disk is divided into smaller units, which help organize how it stores and retrieves information. Major subdivisions are:

Sectors:

The smallest units of division on a disk are called sectors. Each sector is normally either 512 bytes or 4 KB in size. Data is read and written into the disk in such small things.

Clusters:

A cluster is a collection of sectors considered as a single unit for the purpose of keeping files. File systems contain clusters upon which files are stored. A file may span over several clusters when the size of the file is greater than that of a single cluster.

Relationship Between Sectors, Clusters, and Tracks:

Track: Multiple sectors make a track and one or more sectors make a cluster. When a file is written on the disk, OS allocates one or more cluster which used for storing data through contiguous sectors along the tracks.

The read-write head of the disk traverses along the tracks to reach different data sectors. OS manage the physical layout of the files so that data is stored optimally and retrieved quickly by mapping logical file data to physical disk sectors.

Disk Storage and Retrieval:

When a file is accessed, the OS translates the logical block addresses of the file into physical sectors and tracks on the disk for locating the data.

The file allocation table or another similar structure indicates the allocation of clusters on the disk to a file so that the file can be read and written correctly.

Q:12 Discuss the various methods for implementing directories in an operating system, such as linear lists, hash tables, and tree structures. What are the advantages and disadvantages of each method in terms of search efficiency and ease of management?

Methods for Implementing Directories in an Operating System:

Linear Lists:

One type of linear list contains directory entries in a sequential list usually arranged in an array or linked list, where each entry points to a file or subdirectory, giving the file name and recording the disk location.

Pros:

Simplicity: Easy to implement and understand.

Memory Efficient: It saves memory for systems in which users account for a small number of files or directories.

Cons:

Search Efficiency: Since all entries have to be checked sequentially, searching a file is slow, that is, O(n).

Insertion and Deletion: This involves shifting several other elements in the list, which is inefficient and adds to complications.

Hash Tables:

Names associated with a directory entry are converted into unique indices using a particular hash function, through which any file's metadata then becomes directly accessible. At these indices in the array, every file or subdirectory storing the directory entries is chained to counter collisions.

Pros:

Search Efficiency: Searching for files in hash tables is much faster than in linear lists, with an average complexity time of O(1) for hash lookups (assuming good hash).

It has fast insertion/deletion time since the average time complexity for insertion and deletion of files is O(1).

Cons:

Collisions: Collisions happen when a number of directory entries hash to the same index, and although it is possible to handle such collisions using chaining or open addressing, neither method is very appealing in maintaining performance.

Memory Use: Hash tables may require extra memory for the underlying array, particularly if the directory has very few entries.

Number of entries.

Tree Structures:

Directory entries are organized hierarchically and tree structures provide the base for this. Binary search trees (BST), commeune B-trees, provide means to organize entries in a directory. A node in a tree represents the directory or a file, and the child nodes represent files and subdirectories inside that directory. The tree thus arranged can be balanced as per performance optimization with respect to that structure.

Advantages:

Searching: Searching files in a balanced tree yields a time complexity of O(log n) making it much faster than linear lists.

Hierarchical Organization: Natural support for hierarchical organization of directories. It brings much ease in representation and navigation in cases of directories with much subdirectories.

Disadvantages:

Complexity: Tree structures are much more complex to implement and maintain than a linear list or a hash table.

Overhead: More space for storing pointers and managing tree balance, which impacts performance in scenarios such as small directory size.

Q:13 What techniques are used for free space management in an operating system, and how do methods like bitmaps and linked lists help track free and allocated disk space? How do these techniques impact the performance of the file system?

Free Space Management Techniques:

Free space management is vital for efficiently tracking unoccupied disk space in the assumed way to allocate to it when developing new files or data blocks. There are many techniques for denoting allocated and free disk space:

Bitmaps (Bitmap Representation):

In this case, a bitmap uses a bit for each disk block or cluster. Each bit indicates whether the corresponding block is free (0) or allocated (1).

Pros:

Space Efficiency: A bitmap can abstractly represent many blocks very compactly (for example, one bit per block), meaning it takes up less internal memory space.

Quick Search: Searching the next free block is efficient, since the complete bitmap can be traversed all at once.

Management: Bits are just as simple to create and manage from the allocation and deallocation aspect.

Cons:

Overhead: The size of a bitmap increases with the number of blocks in the disk. One can say that this results in a rather heavy memory advantage with very large disks.

Fragmentation: Even though they help with defining and tracking free space, fragmentation may occur when blocks are allocated and freed repeatedly.

Linked Lists: In the linked list method, every free block on the disk contains a pointer to the next free block. The free blocks are linked together like a chain of blocks in this way and allow the file system to find the next available space by following pointers.

Advantages:

No Overhead: Linked lists do not require any additional space for bitmap and they become memory efficient in some instances.

No Fragmentation: Since blocks are allocated in a linked fashion, they can be scattered without getting fragmented into the disk and the organized view of the available blocks could be got from a list.

Disadvantages:

Slower Search: To get the next free block, it requires the traversal of the linked list which makes it low in speed, especially when the list is long.

Pointer Overhead: The extra pointer required for each free block toward the next free block adds up as memory and disk overhead.

Groupings (Block Groups):

Block grouping basically divides the disk into small groups of blocks and each group maintains its own free list or bitmap. Thus, instead of looking through the whole disk, one can restrict the search area for free blocks to smaller sections of the disk.

Advantages:

Reduced Search Time: Such small search spaces prove to substantially reduce the time needed to find free blocks.

Improved Locality: Blocks within the same group are often placed closer together on the disk and yield better performance allocating space for files.

Disadvantages:

Complexity: Free space management algorithms are made more complex with more than one free list or bitmap across all groups.

Uneven Space Allocation: This may lead to certain block groups holding more space than others, thus depriving certain groups of enough space.

Q:14 Explain how the physical disk structure (sectors, tracks, and cylinders) impacts file storage and access. How do disk scheduling algorithms like FCFS, SSTF, and C-SCAN optimize disk access times for a system with multiple processes?

Disk architecture and scheduling Disk Structure The physical disk consists of several layers of data organization, and these layers have direct impact on file storage and access:

Sectors:

These are the smallest units of data storage in a disk which is typically 512 bytes or 4 KB. Each track is divided into several sectors.

Tracks:

Concentric circles that form the disk platter where data are stored. Each platter contains multiple tracks and all tracks on a single platter form the data storage area.

Cylinders:

A number of vertically arranged tracks across all platters in a disk. Data will be from the cylinders when the read/write head is repositioned to appropriate track across platters.

Effect on File Storage and Access

Files are broken down into smaller data blocks which are stored in different sectors: across tracks and cylinders.

The location of data on the disk defines the time it would take for it to be accessed: Seek Time: Time taken by the disk head to get to the appropriate track. Rotational Latency: The amount of time needed to rotate the needed sector under the read/write head as Transfer Time: Time taken for data transfer from the disk to memory. Efficient access to files in turn

occurs when seek times and rotational latencies are minimized, which is exactly the role of disk scheduling algorithms at stake.

Disk Scheduling Algorithms

It is used to service I/O requests in the order determined to optimize the performance of the disk itself:

First Come First Served (FCFS):

Requests will be completed in the order they arrive.

Pros:

Simplicity and fairness.

Cons:

High seek time may result from seeking between long tracks.

Shortest Seek Time First (SSTF):

When service is given to the next closest request to where the head of the disk is positioned.

Pros:

Decreases the average seek time.

Cons:

Starvation of requests farthest from the disk head may occur.

Disk Scheduling Algorithms

Disk Scheduling is the process by which performance is optimized with respect to servicing I/O requests in the order determined with which to service I/O requests.

First Come First Served (FCFS):

It processes requests in the arrival order.

Pros:

Fair and Simple.

Cons:

Can generate considerable seek time due to very long jumps between tracks.

Shortest Seek Time First (SSTF):

Gives priority to the request which is closest to the current location of the disk head.

Pros:

Average seek time will be made less.

Cons:

Starvation of requests farthest from the disk head may occur.

Q:15 What is swap space, and why is it important in virtual memory management? Discuss how the operating system manages swap space, including the role of swapping pages between main memory and secondary storage.

What actually is swap space?

Swap space refers to a reserved portion in secondary storage (like a hard disk or SSD) which functions as an extension of physical memory in the computer (for example, RAM). Swap space is fundamental to the management of virtual memory.

When it comes to virtual memory management, it:

Extends RAM:

It is used when there is low physical memory to keep inactive parts of a process or data temporarily. Additionally, it facilitates multitasking.

This allows running more processes through the system at once than it can actually accommodate in physical memory.

Memory overcommitment is taken care of by the system so that it doesn't stop functioning in the event that the demand for memory exceeds the available RAM.

How Swap Space Functions

Swapping Pages:

Swapping Out:

The operating system shifts less-frequently used pages to swap space when RAM is full.

Swapping In:

When we require a page that is currently in swap space, it is retrieved from swap space and put into RAM.

Page Replacement;

The operating system has defined algorithms such as Least Recently Used (LRU) to determine which pages swap out.

Management of Allocation of Swap Space:

The swap space must be defined when installing the system and is sized according to the needs of the system (for instance, 1.5–2 times the physical RAM for older systems).

Access:

Swap space operations are much slower than RAM due to the disk latency, which is why such use needs to be minimized.

Optimizing:

Maximum utilization of swap space is:

Demand Paging: Swapping out only those pages which are not immediately needed.

Compression: Some systems will compress data before swapping to lessen the disk I/O.

Benefits of Swap Space

Support of Large Applications:

Programs designed to utilize more memory than the available physical memory may be run.

Process Isolation:

Accessing more memory becomes possible without the risk of mutual interference.

System Stability:

Guard against potential crashes by supplementing memory in use.

Inherent Problems With Swap Space

Performance Overhead:

Too much swapping (thrashing) leads to a performance hit of the entire system for that CPU is busy moving data between RAM and disk, rather than executing processes.

Wear of Disks:

The repeated swapping on SSDs would make the storage medium worn out soon.

Q:16 How does an operating system implement system protection mechanisms to prevent unauthorized access to resources and ensure data integrity? Discuss various techniques such as user authentication, access control lists (ACLs), and system calls for protection.

1. User Authentication

Authentication is an action whose outcome is that access to a system can only be managed by authorized users and verified by their identity.

Techniques:

Passwords:

Most popular means.

A unique username and password are provided by users to gain access.

Secure password policies (length, complexity, expiration) further promote this protection.

Biometric Authentication:

Use of biologically unique characteristics like fingerprint, face recognition, or retina scanning.

Two-Factor Authentication (2FA):

Combination of 2 methods like a password and one time code sent to a mobile device.

Single Sign-On (SSO):

Role in Protection:

Prevents unauthorized users from being able to access the system.

Provides accountability through user identification.

2. Access Control Lists (ACLs)

Access control lists (ACLs) define the permissions that a user and the user's process may have to access system resources such as files, directories, or devices.

Components

Subjects: Users or processes requesting access.

Objects: Resources being accessed (like files, memory).

Permissions: Defines what an action allows (like read, write, execute).

How ACLs work:

Every object is associated with a list defining what users or groups are allowed to do to it. Example: File A: User1: Read, Write User2: Read Group1: Execute Role of Protection: Fine-grained access to resources Preventing unauthorized modification and deletion of sensitive data.

Q:17 What are virtual machines (VMs), and how do they enable multiple operating systems to run simultaneously on a single physical machine? Discuss the role of the hypervisor and the distinction between Type 1 and Type 2 hypervisors in virtual machine management.

Definition

VM or Virtual Machine, is a software emulation of an environment providing the functionality of a physical computer, or sometimes it is described as making it possible for multiple operating systems (OS's) to run simultaneously on one physical box making abstraction for the underlying hardware.

How VMs Work

Hardware virtualizations allow the hardware components such as processors, memory, storage, and network interface cards to be emulated in hardware. Each VM has its own independent operating system and applications. Resource Allocation: The virtualization layer is the one dividing the physical resources, for example, CPU, RAM, storage, and assigning each VM with these.

Isolation: VMs are isolated from each other because a failure occurring to one VM will not affect the rest. Benefits of Virtual Machines: Maximal utilization of resources: Allows several VMs to be packed into maximum hardware....

Flexibility: Enables the different software setups or OS to run on identical hardware. Isolation: Each VM runs in its own secured environment. Economically feasible: The number of physical machines required is lesser.

Management made easy: Can clone the VM back up or migrate it, all with minimal downtime.

A hypervisor can also be defined as a Virtual Machine Monitor (VMM. It is software or firmware that virtualizes or creates and manages virtual machines or VMs, distributing hardware resources among them.

Functions of Hypervisor-

Resource Allocation:

Assigns CPU, memory, and storage to VMs as per demand.

Isolation:

VMs run independently without disturbing one another.

Hardware Abstraction:

Abstracts the hardware on which different OSs run on the same physical machine.

Monitoring:

Controls VM performance and optimizes resource utilization.

Types of Hypervisor

1. Type 1 Hypervisor (Bare-Metal Hypervisor)

Definition: A type 1 hypervisor runs directly on physical hardware with no underlying OS.

Examples: VMware ESXi, Microsoft Hyper-V, Xen.

Features:

High efficiency, as there is no middle tier in the OS.

Mostly operate in enterprise settings and/or data centers.

Advantages:

Due to the direct interfacing with hardware, better utilization of resources, and performing work.

More security as has a smaller attack surface being an assembly of code.

Disadvantages:

Demands a specialized hardware and knowledge about it for its setting and management.

2. Type 2 Hypervisor (Hosted Hypervisor)

Definition: Runs on top of an existing host operating system.

Examples: VMware Workstation, Oracle VirtualBox, Parallels Desktop.

Features:

Set up and even used on personal computers.

Development, testing, and educational purposes.

Advantages:

Simple installation and use.

Can function on a standard hardware system with an existing operating system.

Disadvantages:

Performance hits due to the host OS.

Less secure as it depends on how secure the underlying operating system is.

Q:18 What are the key principles of operating system security? Discuss mechanisms such as encryption, access control, auditing, and system updates that protect an OS from threats like malware, unauthorized access, and data breaches.

Security of Operating System

Normally, operating system security encompasses all practices, tools, and mechanisms that are implemented to help safeguard the operating system from malware, unauthorized access, and data breach among others.

The secure operating systems will ensure that data and systems are secured in terms of confidentiality, integrity, and availability.

Fundamental Principles of Operating System Security

Confidentiality:

The implementation of a confidential mechanism grants access to sensitive information to only duly authorized users and processes.

Prevent unauthorized parties from disclosing the data.

Integrity:

The data is kept from being either modified or tampered with unauthorized agents.

It assures that system operations and processes are dependable and reliable.

Availability:

It sends the message of provision to legitimate users without having them wait for resources and services.

It prevents denial of service (DoS) attacks and hardware failures.

Authentication:

The process of user and device verification that allows the access.

Access Control:

It restricts users and processes to only the resources that they are permitted to1 access.

Non-repudiation:

It enables actions within the system to be traceable and verifiable, avoiding denial of responsibility.

Malware Protection

Methods:

Antivirus Software: Scanning and removing malicious code from the computer.

Sandboxing applications to stop them from affecting Malware in systems.

Best Practice:

Regular scans.

Avoidance of untrusted software and sites.

Secure Boot:

only load trusted software into memory at boot up.

How It Works:

Digital signature validation of OS components and drivers.

Benefits:

Protection from root kits and unauthorized modifications.

Firewalls

monitors and controls the traffic between networks according to predefined security rules.

1. Host-based firewalls for the individual system

2. Network firewalls for organizational protection.

User Authentication

Ensuring only legitimate users access the systems.

Methods:

Password Authentication: Strong complex passwords

2FA: Password plus something else

Biometric: Fingerprint, face, iris, etc.