Virtual memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of main memory. The addresses a program may use to reference memory are distinguished from the addresses the memory system uses to identify physical storage sites, and program generated addresses are translated automatically to the corresponding machine addresses.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by a program, called virtual addresses, into physical addresses in computer memory.

1. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of main memory such that it occupies different places in main memory at different times during the course of execution.
2. A process may be broken into number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run-time address translation and use of page or segment table permits this.

If these characteristics are present then, it is not necessary that all the pages or segments are present in the main memory during execution. This means that the required pages need to be loaded into memory whenever required. Virtual memory is implemented using Demand Paging or Demand Segmentation.

Demand paging

The process of loading the page into memory on demand (whenever page fault occurs) is known as demand paging.  
The process includes the following steps :

1. If CPU try to refer a page that is currently not available in the main memory, it generates an interrupt indicating memory access fault.
2. The OS puts the interrupted process in a blocking state. For the execution to proceed the OS must bring the required page into the memory.
3. The OS will search for the required page in the logical address space.
4. The required page will be brought from logical address space to physical address space. The page replacement algorithms are used for the decision making of replacing the page in physical address space.
5. The page table will updated accordingly.
6. The signal will be sent to the CPU to continue the program execution and it will place the process back into ready state.

Hence whenever a page fault occurs these steps are followed by the operating system and the required page is brought into memory.

**Advantages :**

* More processes may be maintained in the main memory: Because we are going to load only some of the pages of any particular process, there is room for more processes. This leads to more efficient utilization of the processor because it is more likely that at least one of the more numerous processes will be in the ready state at any particular time.
* A process may be larger than all of main memory: One of the most fundamental restrictions in programming is lifted. A process larger than the main memory can be executed because of demand paging. The OS itself loads pages of a process in main memory as required.
* It allows greater multiprogramming levels by using less of the available (primary) memory for each process.

Page fault A page fault occurs when a program attempts to access data or code that is in its address space, but is not currently located in the system RAM.

Copy-n-write Is a resource-management technique used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming) to efficiently implement a "duplicate" or "copy" operation on modifiable resources.[[3]](https://en.wikipedia.org/wiki/Copy-on-write#cite_note-Linux-3) If a resource is duplicated but not modified, it is not necessary to create a new resource; the resource can be shared between the copy and the original. Modifications must still create a copy, hence the technique: the copy operation is deferred to the first write. By sharing resources in this way, it is possible to significantly reduce the resource consumption of unmodified copies, while adding a small overhead to resource-modifying operations.

allows both parent and child processes to initially share the same pages in memory. If either process modifies a shared page, only then is the page copied

Page replacement algorithms In a operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

FIFO, Optimal Page Replacement, LRU …

Victim frame in a page replacement algorithm if there is no free frame in which to put the new page there is a need to decide which existing frame will be replaced. This one is called the victim frame.

FIFO This is the simplest page replacement algorithm. In this algorithm, operating system keeps track of all pages in the memory in a queue, oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

LRU In Least Recently Used (LRU) algorithm is a Greedy algorithm where the page to be replaced is least recently used. The idea is based on locality of reference, the least recently used page is not likely.

Trashing At any given time, only few pages of any process are in main memory and therefore more processes can be maintained in memory. Furthermore time is saved because unused pages are not swapped in and out of memory. However, the OS must be clever about how it manages this scheme. In the steady state practically, all of main memory will be occupied with process’s pages, so that the processor and OS has direct access to as many processes as possible. Thus when the OS brings one page in, it must throw another out. If it throws out a page just before it is used, then it will just have to get that page again almost immediately. Too much of this leads to a condition called Thrashing. The system spends most of its time swapping pages rather than executing instructions. So a good page replacement algorithm is required.  
  
In the given diagram, initial degree of multi programming upto some extent of point(lamda), the CPU utilization is very high and the system resources are utilized 100%. But if we further increase the degree of multi programming the CPU utilization will drastically fall down and the system will spent more time only in the page replacement and the time taken to complete the execution of the process will increase. This situation in the system is called as thrashing.  
  
**Causes of Thrashing :**

1. **High degree of multiprogramming**: If the number of processes keeps on increasing in the memory than number of frames allocated to each process will be decreased. So, less number of frames will be available to each process. Due to this, page fault will occur more frequently and more CPU time will be wasted in just swapping in and out of pages and the utilization will keep on decreasing.

For example:  
Let free frames = 400  
**Case 1**: Number of process = 100  
Then, each process will get 4 frames.

**Case 2**: Number of process = 400  
Each process will get 1 frame.  
Case 2 is a condition of thrashing, as the number of processes are increased, frames per process are decreased. Hence CPU time will be consumed in just swapping pages.

1. **Lacks of Frames**: If a process has less number of frames then less pages of that process will be able to reside in memory and hence more frequent swapping in and out will be required. This may lead to thrashing. Hence sufficient amount of frames must be allocated to each process in order to prevent thrashing.

**Recovery of Thrashing :**

* Do not allow the system to go into thrashing by instructing the long term scheduler not to bring the processes into memory after the threshold.
* If the system is already in thrashing then instruct the mid term schedular to suspend some of the processes so that we can recover the system from thrashing.

Working Set model the collection of pages a process is using actively, and which must thus be memory-resident to prevent this process from thrashing.

If the sum of all working sets of all runnable threads exceeds the size of memory, then stop running some of the threads for a while.

Divide processes into two groups: active and inactive:

When a process is active its entire working set must always be in memory: never execute a thread whose working set is not resident.

When a process becomes inactive, its working set can migrate to disk.

Threads from inactive processes are never scheduled for execution.

The collection of active processes is called the balance set.

The system must have a mechanism for gradually moving processes into and out of the balance set.

As working sets change, the balance set must be adjusted.

Memory Mapped File is a segment of [virtual memory](https://en.wikipedia.org/wiki/Virtual_memory) that has been assigned a direct byte-for-byte correlation with some portion of a file or file-like resource. This resource is typically a file that is physically present on disk, but can also be a device, shared memory object, or other resource that the [operating system](https://en.wikipedia.org/wiki/Operating_system) can reference through a [file descriptor](https://en.wikipedia.org/wiki/File_descriptor). Once present, this correlation between the file and the memory space permits applications to treat the mapped portion as if it were [primary memory](https://en.wikipedia.org/wiki/Primary_memory).

The benefit of memory mapping a file is increasing I/O performance, especially when used on large files. For small files, memory-mapped files can result in a waste of [slack space](https://en.wikipedia.org/wiki/Slack_space)[[1]](https://en.wikipedia.org/wiki/Memory-mapped_file#cite_note-1) as memory maps are always aligned to the page size, which is mostly 4 KiB. Therefore, a 5 KiB file will allocate 8 KiB and thus 3 KiB are wasted. Accessing memory mapped files is faster than using direct read and write operations for two reasons. Firstly, a system call is orders of magnitude slower than a simple change to a program's local memory. Secondly, in most operating systems the memory region mapped actually is the kernel's [page cache](https://en.wikipedia.org/wiki/Page_cache) (file cache), meaning that no copies need to be created in user space.

Different types of masstorage systems. (Magnetic, solid, hard…)

Magnetic Disks: A magnetic disk is a storage device that uses a magnetization process to write, rewrite and access data. It is covered with a magnetic coating and stores data in the form of tracks, spots and sectors. Hard disks, zip disks and floppy disks are common examples of magnetic disks. provide bulk of secondary storage of modern computers. **Positioning time** (**random-access time**) is time to move disk arm to desired cylinder (**seek time**) and time for desired sector to rotate under the disk head (**rotational latency**). **Access Latency** = **Average access time** = average seek time + average latency.

Solid State Disks (SSDs): is a [nonvolatile](https://searchstorage.techtarget.com/definition/nonvolatile-memory) storage device that stores persistent [data](https://searchdatamanagement.techtarget.com/definition/data) on [solid-state](https://whatis.techtarget.com/definition/solid-state) [flash memory](https://searchstorage.techtarget.com/definition/flash-memory). Solid-state drives actually aren't hard drives in the traditional sense of the term, as there are no moving parts involved. A traditional hard disk drive ([HDD](https://searchstorage.techtarget.com/definition/hard-disk-drive)) consists of a spinning disk with a read/write head on a mechanical arm called an [actuator](https://internetofthingsagenda.techtarget.com/definition/actuator). An SSD, on the other hand, has an [array](https://searchstorage.techtarget.com/definition/array) of [semiconductor](https://whatis.techtarget.com/definition/semiconductor) memory organized as a disk drive, using integrated circuits ([ICs](https://whatis.techtarget.com/definition/integrated-circuit-IC)) rather than [magnetic](https://whatis.techtarget.com/definition/magnetic-storage) or [optical storage](https://searchstorage.techtarget.com/definition/optical-storage) media

Disk Scheduling and DS algorithms.

Disk scheduling is is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling.

Disk scheduling is important because:

Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by disk controller. Thus other I/O requests need to wait in waiting queue and need to be scheduled.

Two or more request may be far from each other so can result in greater disk arm movement.

Hard drives are one of the slowest parts of computer system and thus need to be accessed in an efficient manner.

There are many Disk Scheduling Algorithms but before discussing them let’s have a quick look at some of the important terms:

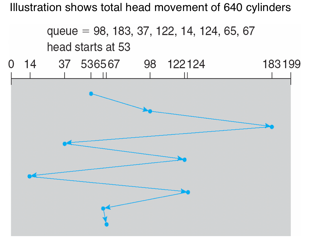
Seek Time:Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.

Rotational Latency: Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.

Transfer Time: Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.

Disk Access Time: Disk Access Time is: seek time + rotational latency + transfer time

FCFS: FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.

Advantages:

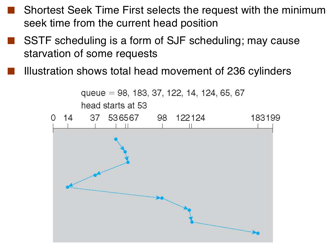
Every request gets a fair chance

No indefinite postponement

Disadvantages:

Does not try to optimize seek time

May not provide the best possible service

SSTF: In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS as it decreases the average response time and increases the throughput of system.

Advantages:

Average Response Time decreases

Throughput increases

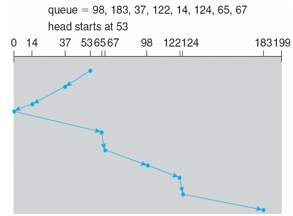
Disadvantages:

Overhead to calculate seek time in advance

Can cause Starvation for a request if it has higher seek time as compared to incoming requests

High variance of response time as SSTF favours only some requests

SCAN: In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence also known as elevator algorithm. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages:

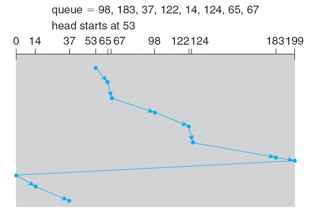
High throughput

Low variance of response time

Average response time

Disadvantages:

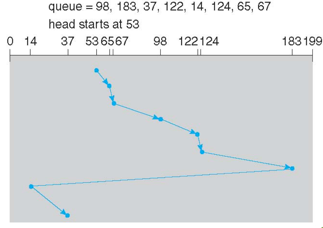
Long waiting time for requests for locations just visited by disk arm

CSCAN: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in CSAN algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

Advantages:

Provides more uniform wait time compared to SCAN

LOOK: It is similar to the SCAN disk scheduling algorithm except the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk. No pic.

CLOOK: As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm inspite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end’s last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

Disk Management

Low-level formatting, or physical formatting — Dividing a disk into sectors that the disk controller can read and write

To use a disk to hold files, the operating system still needs to record its own data structures on the disk

Partition the disk into one or more groups of cylinders, each treated as a logical disk

Logical formatting or “making a file system”

To increase efficiency most file systems group blocks into clusters

Disk I/O done in blocks

File I/O done in clusters

Raw disk access for apps that want to do their own block management, keep OS out of the way

Boot block initializes system

The bootstrap is stored in ROM

Bootstrap loader program stored in boot blocks of boot partition

Methods such as sector sparing used to handle bad blocks

Partition is a logical division of a [hard disk](https://searchstorage.techtarget.com/definition/hard-disk) that is treated as a separate unit by operating systems ([OSes](https://whatis.techtarget.com/definition/operating-system-OS)) and [file systems](https://searchstorage.techtarget.com/definition/file-system). The OSes and file systems can manage information on each partition as if it were a distinct [hard drive](https://searchstorage.techtarget.com/definition/hard-disk-drive). This allows the drive to operate as several smaller sections to improve efficiency, although

it reduces usable space on the hard disk.

Bootstrap Loader is a [program](https://www.computerhope.com/jargon/p/program.htm) that resides in the computer's [EPROM](https://www.computerhope.com/jargon/e/eprom.htm), [ROM](https://www.computerhope.com/jargon/r/rom.htm), or other [non-volatile memory](https://www.computerhope.com/jargon/m/memory.htm). It is automatically executed by the processor when turning on the computer. The bootstrap loader reads the [hard drives](https://www.computerhope.com/jargon/h/harddriv.htm) boot sector to continue the process of loading the computer's [operating system](https://www.computerhope.com/jargon/o/os.htm).

RAID structure or “Redundant Arrays of Inexpensive Disks” is a technique which makes use of a combination of multiple disks instead of using a single disk for increased performance, data redundancy or both.

Key evaluation points for a RAID System

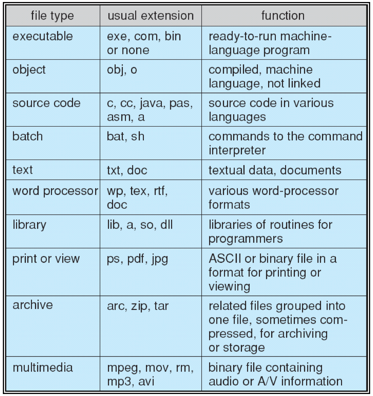
Reliability: How many disk faults can the system tolerate?

Availability: What fraction of the total session time is a system in uptime mode, i.e. how available is the system for actual use?

Performance: How good is the response time? How high is the throughput (rate of processing work)? Note that performance contains a lot of parameters and not just the two.

Capacity: Given a set of N disks each with B blocks, how much useful capacity is available to the user?

RAID is very transparent to the underlying system. This means, to the host system, it appears as a single big disk presenting itself as a linear array of blocks. This allows older technologies to be replaced by RAID without making too many changes in the existing code.

File concept is a contiguous logical address space. Can be of different types (Data[numeric, character, binary] or Program).

File Attributes **Name** (kept in human readable form), **Identifier** (unique tag number to identify it within the file system), **Type** (needed if the system supports different types), **Location** (pointer to it’s location), **Size, Protection** (who can read write and execute), **Time date and user identification** (for protection and security)

File Operations **create, write, read, reposition within file, delete, truncate, Open ( se**arch the directory structure on disk for the entry and move it’s contents to memory**.), Close (**move entry’s content from memory to disk**)**

Files Structures: **none** (sequence of words/bytes), **Simple record structure** (lines/fixed length, variable length), **Complex structures** (formattable document, relocatable load file)

(sequential access, random access…)

Access File Methods refers to the manner in which the records of a file may be accessed. There are several ways to access files: Sequential access. /. Direct/Random access /. Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

Random access file organization provides, accessing the records directly.

Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.

The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

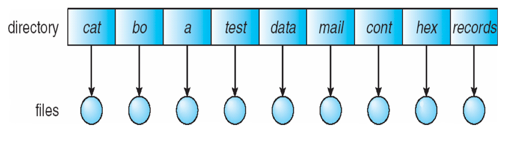
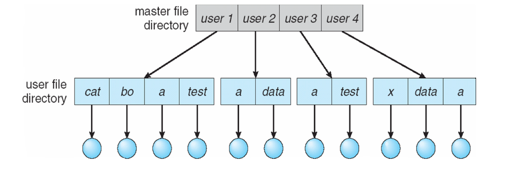
This mechanism is built up on base of sequential access.

An index is created for each file which contains pointers to various blocks.

Index is searched sequentially and its pointer is used to access the file directly.

Directory: Collection of files is a file directory. The directory contains information about the files, including attributes, location and ownership. Much of this information, especially that is concerned with storage, is managed by the operating system. The directory is itself a file, accessible by various file management routines.

Information contained in a device directory are:

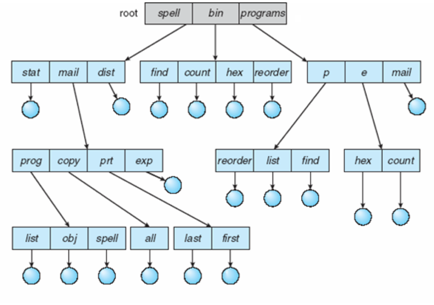
* ****Name
* Type
* Address
* Current length
* Maximum length
* Date last accessed
* Date last updated
* Owner id
* Protection information

Operation performed on directory are:

* Search for a file
* Create a file
* Delete a file
* List a directory
* Rename a file
* Traverse the file system

Advantages of maintaining directories are:

* Efficiency: A file can be located more quickly.
* Naming: It becomes convenient for users as two users can have same name for different files or may have different name for same file.
* Grouping: Logical grouping of files can be done by properties e.g. all java programs, all games etc.

****Directory Organization

**SINGLE-LEVEL DIRECTORY**

In this a single directory is maintained for all the users.

* Naming problem: Users cannot have same name for two files.
* Grouping problem: Users cannot group files according to their need.

**TWO-LEVEL DIRECTORY**

In this separate directories for each user is maintained.

Path name: Due to two levels there is a path name for every file to locate that file.

Now, we can have same file name for different user.

Searching is efficient in this method.

**TREE-STRUCTURED DIRECTORY :**

Directory is maintained in the form of a tree.

Searching is efficient and also there is grouping capability.

We have absolute or relative path name for a file.

Acyclic Graph

Allow directories to link to one another, allow multiple directories to contain same file i.e., only one copy of the file exists and any change in the file can be viewed by all directories in which it is contained.

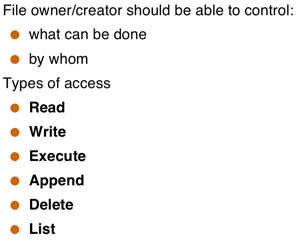
Results in Acyclic graphs

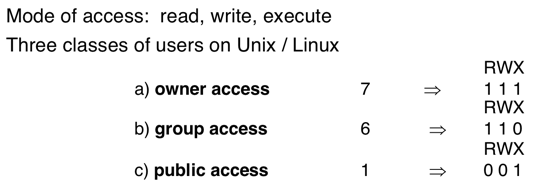
Two users can name same file or same directory

Duplicate paths may complicate task of backing up periodically

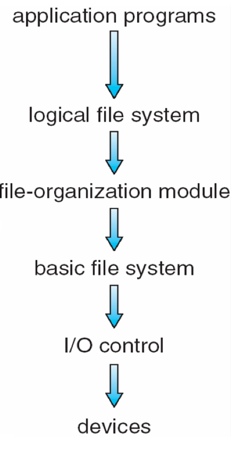
File System Mounting. Before you can access the files on a file system, you need to mount the file system. Mounting a file system attaches that file system to a directory (mount point) and makes it available to the system. The root (/) file system is always mounted. Any other file system can be connected or disconnected from the root (/) file system.

File Protection





Layered File System. Benefits, what for?

**Logical file system**  
Provides users the view of a contiguous sequence of words, bytes stored somewhere.  
Uses a directory structure, symbolic name  
Provides protection and security  
OS/user interface

The f**ile organization module** Knows about files and their logical blocks (say 1,..N) Files are organized in blocks of 32 bytes to 4K bytes. Translates logical blocks into physical .Knows location of file, file allocation type. Includes a free space manages that tracks unallocated blocks

**Basic file system**  
Issues commands to the device driver (layer of software that directly controls disk hardware) to read and write physical blocks on the disk,  
Each physical block identified by a disk address (e.g., drive 2, cylinder 34, track 2, sector 11)

**IO control**  
The lowest level in the file system  
Consists of device drivers and interrupt handlers to transfer information between the memory and the disk

A **device** driver translates commands such as “get me block 111” into hardware specific ISA used by hardware controller. This is accomplished by writing specific bits into IO registers.

File control Block FCB is a file system structure in which the state of an open [file](https://en.wikipedia.org/wiki/Computer_file) is maintained. A FCB is managed by the operating system, but it resides in the memory of the program that uses the file, not in operating system memory. This allows a process to have as many files open at one time as it wants to, provided it can spare enough memory for an FCB per file.

How would a directory be implemented

File-Allocation Table.

Allocation Methods The allocation methods define how the files are stored in the disk blocks

**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.

**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 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.

**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.

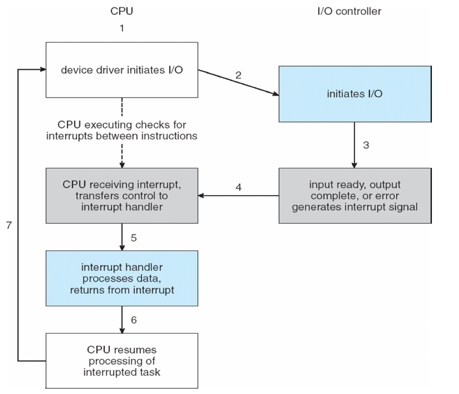
Disadvantages:

* 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.

Free-Space Management.  To keep track of free disk space, the system maintains a free-space list. The free-space list records all free disk blocks—those not allocated to some file or directory. To create a file, we search the free-space list for the required amount of space and allocate that space to the new file. This space is then removed from the free-space list.

Page Cache is a transparent [cache](https://en.wikipedia.org/wiki/Cache_(computing)) for the [pages](https://en.wikipedia.org/wiki/Page_(computer_memory)) originating from a [secondary storage](https://en.wikipedia.org/wiki/Secondary_storage) device such as a [hard disk drive](https://en.wikipedia.org/wiki/Hard_disk_drive) (HDD). The [operating system](https://en.wikipedia.org/wiki/Operating_system) keeps a page cache in otherwise unused portions of the [main memory](https://en.wikipedia.org/wiki/Main_memory) (RAM), resulting in quicker access to the contents of cached pages and overall performance improvements. A page cache is implemented in [kernels](https://en.wikipedia.org/wiki/Kernel_(computer_science)) with the [paging](https://en.wikipedia.org/wiki/Paging) memory management, and is mostly transparent to applications.

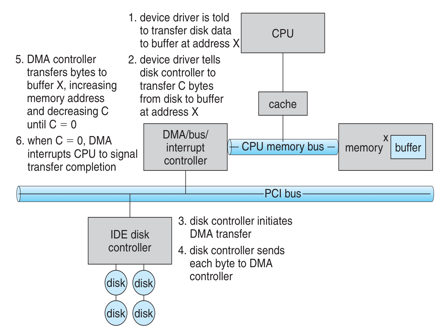
NFS is a [distributed file system](https://en.wikipedia.org/wiki/Distributed_file_system) protocol originally developed by [Sun Microsystems](https://en.wikipedia.org/wiki/Sun_Microsystems) in 1984,[[1]](https://en.wikipedia.org/wiki/Network_File_System#cite_note-sun85-1) allowing a user on a client [computer](https://en.wikipedia.org/wiki/Computer) to access files over a [computer network](https://en.wikipedia.org/wiki/Computer_network) much like local storage is accessed

Memory Mapped I/O method of performing [input/output](https://en.wikipedia.org/wiki/Input/output) (I/O) between the [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit) (CPU) and [peripheral devices](https://en.wikipedia.org/wiki/Peripheral_device) in a [computer](https://en.wikipedia.org/wiki/Computer). Memory-mapped I/O uses the same [address space](https://en.wikipedia.org/wiki/Address_space) to address both [memory](https://en.wikipedia.org/wiki/Computer_memory) and [I/O devices](https://en.wikipedia.org/wiki/I/O_device). The memory and [registers](https://en.wikipedia.org/wiki/Register_(computing)) of the I/O devices are mapped to (associated with) address values. So when an address is accessed by the CPU, it may refer to a portion of [physical RAM](https://en.wikipedia.org/wiki/Physical_memory), or it can instead refer to memory of the I/O device. Thus, the CPU instructions used to access the memory can also be used for accessing devices.

Direct I/O For applications that wish to bypass the buffering of memory within the file system cache, Direct I/O is provided. When Direct I/O is used for a file, data is transferred directly from the disk to the application buffer, without the use of the file buffer cache.

Polling (pros and cons) Polling is the process where the computer or controlling device waits for an [external device](https://en.wikipedia.org/wiki/External_device) to check for its readiness or state. Can be too inefficient if the device is slow.

Interrupt Cycle 🡪

Direct Memory Access. Benefits. Disadvantages?  is a feature of computer systems that allows certain hardware subsystems to access main system [memory](https://en.wikipedia.org/wiki/Computer_storage) ([Random-access memory](https://en.wikipedia.org/wiki/Random-access_memory)), independent of the [central processing unit](https://en.wikipedia.org/wiki/Central_processing_unit) (CPU). Without DMA, when the CPU is using [programmed input/output](https://en.wikipedia.org/wiki/Programmed_input/output), it is typically fully occupied for the entire duration of the read or write operation, and is thus unavailable to perform other work. With DMA, the CPU first initiates the transfer, then it does other operations while the transfer is in progress, and it finally receives an [interrupt](https://en.wikipedia.org/wiki/Interrupt) from the DMA controller when the operation is done. This feature is useful at any time that the CPU cannot keep up with the rate of data transfer, or when the CPU needs to perform work while waiting for a relatively slow I/O data transfer.

Slide 15 🡪

Block and Character Devices differences. Include disk drives vs include keyborads, mice, serial ports…

Blocking vs Nonblocking vs Asynchronous I/O

**Blocking** process suspend until I/O completion

**Nonblocking** I/O calls returns as much as possible.

**Asynchronou**s process run while I/O executes

Kernel I/O Subsystem

is responsible to provide many services related to I/O. Following are some of the services provided.

Scheduling − Kernel schedules a set of I/O requests to determine a good order in which to execute them. When an application issues a blocking I/O system call, the request is placed on the queue for that device. The Kernel I/O scheduler rearranges the order of the queue to improve the overall system efficiency and the average response time experienced by the applications.

Buffering − Kernel I/O Subsystem maintains a memory area known as buffer that stores data while they are transferred between two devices or between a device with an application operation. Buffering is done to cope with a speed mismatch between the producer and consumer of a data stream or to adapt between devices that have different data transfer sizes.

Caching − Kernel maintains cache memory which is region of fast memory that holds copies of data. Access to the cached copy is more efficient than access to the original.

Spooling and Device Reservation − A spool is a buffer that holds output for a device, such as a printer, that cannot accept interleaved data streams. The spooling system copies the queued spool files to the printer one at a time. In some operating systems, spooling is managed by a system daemon process. In other operating systems, it is handled by an in kernel thread.

Error Handling − An operating system that uses protected memory can guard against many kinds of hardware and application errors.

Goals of Protection

Fines grained…

Domain Structure

Access Matrix. Use of it.

Access Control

Security Violation Categories

Security Violation Methods

Security Measure Levels

Program Threats

Buffer Overflow

Buffer Overflow code fix \*\*\*\*

FIFO IN C++ LRU C++

|  |
| --- |
|  |