



OPERATING SYSTEMS

UE19CS254 Unit 4 Revision Class #1

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

OPERATING SYSTEMS

Course Syllabus - Unit 4



Unit 4: Storage Management

Mass-Storage Structur - Mass-Storage overview, Disk Scheduling, Swap-Space Management, RAID structure. File System Interface - file organization/structure and access methods, directories, sharing File System Implementation/Internals: File control Block (inode), partitions & mounting, Allocation methods.

Case Study: Linux/Windows File Systems

OPERATING SYSTEMS

Course Outline



37	Mass-Storage Structure: Mass-Storage overview	12.1	82.1
38	Disk Scheduling – FCFS, SSTF, SCAN, C-SCAN, LOOK	12.4	
39	Swap-Space Management, RAID Structure	12.6,12.7	
40	File Concept, File Structure, Access Methods	10.1-10.2	
41	Directory and Disk Structure	10.3	
42	File-System Mounting, File Sharing, Protecting	10.4-10.6	
43	Implementing File-Systems: File control Block (inode), partitions & mounting	11.1,11.2	
44	Disk Space Allocation methods: Contiguous, Linked, Indexed	11.4	
45	Case Study: Unix/Linux File systems	16.7	
46	NFS	11.8	

OPERATING SYSTEMS

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

blocks	a unit of a specific size, that is used for I/O transfers between memory and disk
File systems	provide efficient and convenient access to the disk by allowing data to be stored, located, and retrieved easily.
I/O control	consists of device drivers and interrupt handlers to transfer information between the main memory and the disk system.
Basic File System	issues generic commands to the appropriate device driver to read and write physical blocks on the disk.
file-organization module	understands files, logical address, and physical blocks. translates logical block addresses to physical block addresses for the basic file system to transfer.
logical file system	manages metadata information
file-control block (FCB) (inode)	contains information about the file, including ownership, permissions, and location of the file contents.

UNIX file system (ufs)	Primary file system for Unix and Unix based operating systems that uses a hierarchical file system structure where the highest level of the directory called root (/ pronounced "slash") and all other directories span from that root
extended file system	the standard Linux file system
File System Implementation	<p>The Structures and operations used to implement file system operations. Included:</p> <ul style="list-style-type: none">* Boot control block* volume control block* a directory file structure* A per file FCB
boot control block	contains information needed by the system to boot an operating system from a volume. Also called boot block and partition boot sector
Volume control block	contains volume (or partition) details, such as the number of blocks in the partition, the size of the blocks, a free-block count and free-block pointers, and a free-FCB count and FCB pointers. Also called superblock, or master file table
in-memory mount table	contains information about each mounted volume.

OPERATING SYSTEMS

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

system-wide open-file table	contains a copy of the FCB of each open file, as well as other information.
per-process open-file table	contains a pointer to the appropriate entry in the system-wide open-file table, as well as other information
file descriptor (file handle)	information kept in the directory to describe a file or file extent.
raw disk	partition type used when no file system is appropriate
boot loader	An operating system program responsible for managing the process of loading the OS during the boot.
dual boot	The ability to boot your computer using one of two different operating systems.
Root Partition	Contains the operating-system kernel and sometimes other system files, is mounted at boot time.
virtual file system (VFS)	separates file-system-generic operations from their implementation, provides a mechanism for uniquely representing a file throughout a network.
vnode	file-representation structure That contains a numerical designator for a network-wide unique file

inode object	represents an individual file
file object	A value that represents an open file.
superblock object	represents an entire file system
dentry object	represents an individual directory entry
Contiguous Allocation	requires that each file occupy a set of contiguous blocks on the disk.
Linear List	linear list of file names with pointers to the data blocks.
Hash Table	data structure used for a file directory is a hash table. Here, a linear list stores the directory entries, but a hash data structure is also used
Dynamic storage allocation problem	The problem of how to satisfy a request of size n from a list of free holes.
External Fragmentation	As files are allocated and deleted, the free disk space is broken into little pieces. External fragmentation exists whenever free space is broken into chunks.
compact	compacting the current free storage into one contiguous space

extent	a chunk of contiguous space as a segment with pointers to the next chunk of contiguous space
linked allocation	each file is a linked list of blocks scattered anywhere on the disk
cluster	collection of multiple blocks for hard drive use
File Allocation Table (FAT)	method of disk-space allocation was used by the MS-DOS operating system.
Indexed allocation	Each file has an index that contains location pointers for each piece of the file
index block	the link pointers for a file are all stored together in one block
direct block	block which holds file data

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Free Space Management	File system maintains free space list to track available blocks/clusters
free-space list	Records all free disk blocks (those not allocated to some file or directory)
bit map (bit vector)	Each block is represented by 1 bit. If the block is free, the bit is 1; if the block is allocated, the bit is 0.
ZFS	file system (found in Solaris and other operating systems) was designed to encompass huge numbers of files, directories, and even file systems (in ZFS, we can create file-system hierarchies).
metaslabs	slab that divide the space on the device into chunks of manageable size
buffer cache	where blocks are kept under the assumption that they will be used again shortly.
page cache	caches pages rather than disk blocks using virtual memory techniques and addresses
Unified virtual memory	using page caching to cache both process pages and file data
unified buffer cache	contains the same pages for memory-mapped IO as well as ordinary IO

double caching	contents of the file in the buffer cache copied into the page cache. Not only does it waste memory but it also wastes significant CPU and I/O cycles
Synchronous writes	Occur in the order in which the disk subsystem receives them, and the writes are not buffered.
asynchronous write	the data are stored in the cache, and control returns to the caller.
Free-behind	removes a page from the buffer as soon as the next page is requested
read ahead	a requested page and several subsequent pages are read and cached
consistency checker	compares the data in the directory structure with the data blocks on disk and tries to fix any inconsistencies it finds

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

transaction	set of operations for performing a specific task
circular buffer	writes to the end of its space and then continues at the beginning, overwriting older values as it goes.
snapshot	is a view of the file system before the last update took place.
back up	make a copy of (a computer file) especially for storage in another place
restoring	process of retrieving data from a backup storage device
full backup	Backup that copies all data from a system.
incremental backup	A type of partial backup that involves copying only the data items that have changed since the last full backup
Network File System (NFS)	both an implementation and a specification of a software system for accessing remote files across LANs (or even WANs)
NFS protocol	a protocol for remote file accesses
mount protocol	establishes the initial logical connection between a server and a client

OPERATING SYSTEMS

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Hard disks	provide the bulk of secondary storage for modern computer systems
platter	A thin, round, metallic storage plate stacked onto the hard drive spindle.
disk arm	moves all the read-write heads in a harddrive unit
tracks	Circles on a magnetic storage device where data is stored or retrieved.
sectors	sectors on magnetic media used for storing digital information.
cylinder	The set of concentric tracks on all surfaces
Rotations per minute (RPM)	Denotes how fast the platters spin.
transfer rate	the rate at which data flow between the drive and the computer.
Positioning time (random-access time)	Time to move disk arm to desired cylinder (seek time) and time for desired sector to rotate under the disk head (rotational latency)
seek time	the time it takes for a read/write head to move to a specific data track

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

rotational latency	The time necessary for the desired sector to rotate to the disk head
head crash	a hard disk crash caused by the heads coming in contact with the spinning disk(s).
flash drives	are small, very portable memory devices. They come in varying sizes and can be used to transfer information from computer to computer or even directly to some printers.
I/O bus	<p>Input/ Output bus connects the expansion slots (ports) to Processor</p> <p>advanced technology attachment (ATA), serial ATA (SATA), eSATA, universal serial bus (USB), and fibre channel (FC)</p>
controllers	control the data transfers on a bus
host controller	The controller at the computer end of the bus
disk controller	controller built into each hard disk
solid-state disks	nonvolatile memory that is used like a hard drive. Fast
Magnetic tape	relatively permanent and can hold large quantities of data, its access time is slow compared with that of main memory and hard disks

logical blocks	smallest unit of transfer
low-level formatting (physical formatting)	A process (usually performed at the factory) that electronically creates the hard drive tracks and sectors and tests for bad spots on the disk surface.
constant linear velocity (CLV)	the density of bits per track is uniform, speed changes
constant angular velocity (CAV)	the density of bits decreases from inner tracks to outer tracks to keep the data rate constant
host-attached storage	accessed via local I/O ports
Network Attached Storage (NAS)	a network-connected computer dedicated to providing file-based data storage services to other network devices.
Storage Area Network (SAN)	a high-speed network with the sole purpose of providing storage to other attached servers
iSCSI	it uses the IP network protocol to carry the SCSI protocol, allowing for networks to be used to connect storage as if it were directly attached
Disk Scheduling	the act of deciding which outstanding requests for disk I/O to satisfy first
disk bandwidth	total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

shortest-seek-time-first (SSTF) algorithm	selects the request with the least seek time from the current head position. In other words, SSTF chooses the pending request closest to the current head position.
SCAN algorithm (elevator algorithm)	scans from beginning to end for the data requested
Circular SCAN (C-SCAN) Scheduling	variant of SCAN designed to provide a more uniform wait time. Like SCAN, C-SCAN moves the head from one end of the disk to the other, servicing requests along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing any requests on the return trip
Look Scheduling	A Variant of SCAN and CSCAN that ends the disk trip at the end of the last request and scans or returns back to the beginning
Disk Management	Initializes disks, creates partitions, and formats partitions
error-correcting code (ECC)	validates sector mismatches for hard disks

OPERATING SYSTEMS

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

partition	separation or grouping of hard disk cylinders
logical formatting	The creation of a file system
clusters	blocks of data grouped together in larger chunks
raw disk	partition type used when no file system is appropriate
bootstrap	the first program loaded from ROM by BIOS; loads operating system from secondary storage
read-only memory (ROM)	Permanent storage; instructions are burned onto chips by the manufacturer.
boot disk (system disk)	A disk that has a boot partition
boot partition	The hard drive partition where the OS is stored.
Master Boot Record (MBR)	In Windows The first sector on a hard drive, which contains the partition table and a program the BIOS uses to boot an OS from the drive.
bad blocks	The areas of a storage medium unable to store data properly.
sector sparing (forwarding)	tracks bad blocks

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Hard errors	an error that results in loss of data on the hard disk
Swap space management	The main goal for the design and implementation of swap space is to provide the best throughput for the virtual memory system
raw partition	partition where no system or directory structure is placed; used for swap space
page slots	area used to hold swapped pages
swap map	an array of integer counters, each corresponding to a page slot in the swap area.
redundant arrays of independent disks (RAID)	involves using parallel disks that contain redundant elements of data and applications. If one disk fails, the lost data are automatically reconstructed from the redundant components stored on the other disks.
Mean time to failure (MTTF)	The average amount of time expected until the first failure of a piece of equipment.
redundancy	storage of extra information that is not normally needed but that can be used in the event of failure of a disk to rebuild the lost information. Thus, even if a disk fails, data are not lost.

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Mirroring (RAID 1)	With mirroring, a logical disk consists of two physical disks, and every write is carried out on both disks. The result is called a mirrored volume.
Mean Time to Repair (MTTR)	The average amount of time a computer repair technician needs to resolve the cause of a failure through replacement or repair of a faulty unit.
mean time to data loss	The average amount of time of a system is expected to lose data
Nonvolatile RAM (NVRAM)	memory chips that do not lose their contents when the power to the computer is turned off
data striping	A fault-tolerance technique that breaks a unit of data into smaller segments and stores these segments on multiple disks.
bit-level striping	splitting the bits of each byte across multiple disks
block-level striping	blocks of a file are striped across multiple disks
RAID levels	A set of RAID configurations that consists of striping, mirroring, or parity.
read-modify-write cycle	An operating-system write of data smaller than a block requires that the block be read, modified with the new data, and written back. The parity block has to be updated as well

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

P + Q redundancy scheme	stores extra redundant information to guard against multiple disk failures. Instead of parity, error-correcting codes such as the Reed-Solomon codes are used.
snapshot	a view of the file system before the last update took place.
hot spare	A device configured to be used as a replacement in case of disk failure.
checksum	a technique used to verify the integrity of data
inode	The portion of a file that stores information on the file's attributes, access permissions, location, ownership, and file type.
pools (storage)	A pool can hold one or more ZFS file systems. The entire pool's free space is available to all file systems within that pool.

OPERATING SYSTEMS

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

File System	is used to control how data is stored and retrieved.
File System	Without a it, information placed in a storage area would be one large body of data with no way to tell where one piece of information stops and the next begins.
File System	By separating the data into individual pieces, and giving each piece a name, the information is easily separated and identified.
File	Taking its name from the way paper-based information systems are named, each group of data is called a
File System	The structure and logic rules used to manage the groups of information and their names is called a _____
Space management	File systems allocate space in a granular manner, usually multiple physical units on the device.
Space Management	The file system is responsible for organizing files and directories, and keeping track of which areas of the media belong to which file and which are not being used.
File names	is used to identify a storage location in the file system.

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Hard disks secondary storage properties	<p>Hard disks have two important properties that make them suitable for secondary storage of files in file systems:</p> <p>(1) Blocks of data can be rewritten in place, and</p> <p>(2) they are direct access, allowing any block of data to be accessed with only minor movements of the disk heads and rotational latency.</p>
Disks Access	<p>Disks are usually accessed in physical blocks, rather than a <u>byte</u> at a time. Block sizes may range from 512 bytes to 4K or larger</p>
File systems on disk drives layered design	<p>Lowest level physical devices (magnetic media, motors, etc)</p> <p>I/O Control consists of device drivers.</p> <p>Basic file system level</p> <p>File organization module</p> <p>Logical file system</p>
I/O Control layer	<p>consists of device drivers, special software programs (often written in assembly) which communicate with the devices by reading and writing special codes directly to and from memory addresses corresponding to the controller card's registers. Each controller card (device) on a system has a different set of addresses (registers, a.k.a. ports) that it listens to a unique set of command codes and results codes that it understands.</p>

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Basic File System layer	<p>Each physical block is identified by its numeric needs only to issue generic commands to the appropriate device driver to read and write physical blocks on the disk e.g Disk address (for example, drive 1, cylinder 73, track 2, sector 10)</p> <p>Manages the memory buffers and caches that hold various file-system, directory, and data blocks</p> <p>Given command like "retrieve block 123" translates to device driver</p>
File Organization Module layer	<p>Knows about files and their logical blocks, and how they map to physical blocks on the disk.</p> <p>Translating from logical to physical blocks</p> <p>Maintains the list of free blocks, and allocates free blocks to files as needed</p>
Logical File System layer	<p>Deals with all of the meta data associated with a file (UID, GID, mode, dates, etc), i.e. everything about the file except the data itself.</p> <p>Manages the directory structure and the mapping of file names to file control blocks, FCBs, which contain all of the meta data as well as block number information for finding the data on the disk</p>
File systems data structures on the disk	<ul style="list-style-type: none">-boot-control block-volume control block-directory structure-File Control Block, FCB
boot-control block	boot block in UNIX or the partition boot sector in Windows contains information about how to boot the system off of this disk.

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

volume control block	master file table in UNIX or the superblock in Windows, which contains information such as the partition table, number of blocks on each filesystem, and pointers to free blocks and free FCB blocks
directory structure (per file system)	Containing file names and pointers to corresponding FCBs. UNIX uses inode numbers, and NTFS uses a master file table.
File Control Block (FCB)	(per file) containing details about ownership, size, permissions, dates, etc. UNIX stores this information in inodes , and NTFS in the master file table as a relational database structure
File data structures stored in memory	<ul style="list-style-type: none">-An in-memory mount table.-An in-memory directory cache of recently accessed directory information.-A system-wide open file table, containing a copy of the FCB for every currently open file in the system, as well as some other related information.-A per-process open file table, containing a pointer to the system open file table as well as some other information. (For example the current file position pointer may be either here or in the system file table)
Process when a file is created	When a new file is created , a new FCB is allocated and filled out with important information regarding the new file. The appropriate directory is modified with the new file name and FCB information.

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

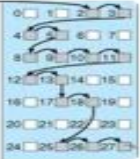
Process when files are accessed	When a file is accessed, the <code>open()</code> system call reads in the FCB information from disk, and stores it in the system-wide open file table . An entry is added to the per-process open file table referencing the system-wide table, and an index into the per-process table is returned by the <code>open()</code> system call. UNIX refers to this index as a file descriptor , and Windows refers to it as a file handle .
Process when another process already has a file open	If another process already has a file open when a new request comes in for the same file, and it is sharable, then a counter in the system-wide table is incremented and the <u>per-process table</u> is adjusted to point to the existing entry in the system-wide table.
Process when file is closed	When a file is closed, the per-process table entry is freed, and the counter in the system-wide table is decremented. If that counter reaches zero, then the system wide table is also freed. Any data currently stored in memory cache for this file is written out to disk if necessary.
UNIX mount point	In UNIX a mount point is indicated by setting a flag in the in-memory copy of the inode , so all future references to that inode get re-directed to the root directory of the mounted filesystem.

VFS Linux	<p>The VFS in Linux is based upon four key object types:</p> <p>The inode object, representing an individual file</p> <p>The file object, representing an open file.</p> <p>The superblock object, representing a filesystem.</p> <p>The dentry object, representing a directory entry.</p>
Directory Implementation	<p>Linear List</p> <p>Hash Table</p>
Directory Implementation - Linear List	<p>simplest and easiest directory structure, Finding a file requires a linear search, Sorting the list makes searches faster, at the expense of more complex insertions and deletions.</p>
Directory Implementation - Hash Table	<p>A hash table can also be used to speed up searches. Hash tables are generally implemented in addition to a linear or other structure</p>
methods of storing files on disks	<p>contiguous allocation</p> <p>linked allocation</p> <p>allocation indexed</p>

Contiguous Allocation	<p>requires that all blocks of a file be kept together contiguously. Problems can arise when files grow, or if the exact size of a file is unknown at creation time:</p> <ul style="list-style-type: none">-Over-estimation of the file's final size increases external fragmentation and wastes disk space.-Under-estimation may require that a file be moved or a process aborted if the file grows beyond its originally allocated space.-If a file grows slowly over a long time period and the total final space must be allocated initially, then a lot of space becomes unusable before the file fills the space.-only one access is needed to get a disk block using direct access
extents	<p>A large contiguous chunks of disk blocks. When a file outgrows its original extent, then an additional one is allocated.</p>
Linked Allocation	<p>Disk files can be stored as linked lists, with the expense of the storage space consumed by each link (no external fragmentation problem). is only efficient for sequential access files, as random access requires starting at the beginning of the list for each new location access.</p> <ul style="list-style-type: none">-Allocating clusters of blocks reduces the space wasted by pointers, at the cost of internal fragmentation.-reliability if a pointer is lost or damaged. Doubly linked lists provide some protection, at the cost of additional overhead and wasted space

File Allocation Table (FAT)	Used by DOS is a variation of linked allocation, where all the links are stored in a separate table at the beginning of the disk. The benefit of this approach is that the FAT table can be cached in memory, greatly improving random access speeds .
Indexed Allocation	<p>combines all of the <i>indexes</i> for accessing each file into a common block (for that file), as opposed to spreading them all over the disk or storing them in a FAT table. Some disk space is wasted (relative to linked lists or FAT tables) because an entire index block must be allocated for each file. So how big the index block should be, and how it should be implemented:</p> <p>Linked Scheme Multi-Level Index Combined Scheme</p> 
Linked Scheme	An index block is one disk block, which can be read and written in a single disk operation. The first index block contains some header information, the first N block addresses, and if necessary a pointer to additional linked index blocks.
Multi-Level Index	The first index block contains a set of pointers to secondary index blocks, which in turn contain pointers to the actual data blocks.

Combined Scheme	<p>This is the scheme used in UNIX inodes, in which the first 12 or so data block pointers are stored directly in the inode, and then singly, doubly, and triply indirect pointers provide access to more data blocks as needed. advantage is that files up to about 4144K (using 4K blocks) are accessible with only a single indirect block (which can be cached)</p> 
Free-Space Management	<p>Another important aspect of disk management is keeping track of and allocating free space. methods:</p> <ul style="list-style-type: none">Bit VectorLinked ListGroupingCountingSpace Maps
Bit Vector	<p>Each bit represents a disk block, set to 1 if free or 0 if allocated.</p> <p>Fast algorithms exist for quickly finding contiguous blocks of a given size</p> <p>The down side is that a 40GB disk requires over 5MB just to store the <i>bitmap</i>. so It is not feasible to keep the entire list in main memory for large disks</p>

Linked List	<p>link together all the free disk blocks, keeping a pointer to the first free block in a special location on the disk and caching it in memory. not efficient (requires substantial I/O time - however, traversing the free list is not a frequent action).</p> 
Grouping	<p>A modification of the free-list approach stores the addresses of n free blocks in the first free block. The first $n-1$ of these blocks are actually free. The last block contains the addresses of another n free blocks, and so on</p>
Counting	<p>generally, several contiguous blocks may be allocated or freed simultaneously, particularly when space is allocated with the contiguous-allocation algorithm or through clustering. Thus, rather than keeping a list of n free disk addresses, we can keep the address of the first free block and the number (n) of free contiguous blocks that follow the first block. Each entry in the free-space list then consists of a disk address and a count.</p>

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Space Maps	<ul style="list-style-type: none">-ZFS uses a combination of techniques, starting with dividing the disk up into (hundreds of) metaslabs of a manageable size, each having their own space map.-Free blocks are managed using the counting technique, but rather than write the information to a table, it is recorded in a log-structured transaction record.-An in-memory space map is constructed using a balanced tree data structure, constructed from the log data-The combination of the in-memory tree and the on-disk log provide for very fast and efficient management of these very large files and free blocks.
performance buffer cache	Some OSes cache disk blocks they expect to need again in a buffer cache .
algorithm for managing buffer cache	Least-recently-used (LRU) algorithm is considered reasonable for managing a buffer cache
Page Cache	uses virtual memory techniques to cache file data as pages as opposed to system-oriented blocks.
performance page cache	A page cache connected to the virtual memory system is actually more efficient as memory addresses do not need to be converted to disk block addresses and back again. Some systems use page caching for both process pages and file data in a unified virtual memory .
unified virtual memory.	use page caching to cache both process pages and file data.

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

unified buffer cache	<ul style="list-style-type: none">- contains the same pages for memory-mapped IO as well as ordinary IO-Data does not need to be stored twice.-Problems of inconsistent buffer information are avoided.
priority paging	giving process pages priority over file <i>I/O pages</i> , and setting limits so that neither can knock the other completely out of memory.
synchronous writes	writes occur in the order in which the disk subsystem receives them, without caching. Metadata writes are often done synchronously.
Asynchronous writes	writes are cached, allowing the disk subsystem to schedule writes in a more efficient order
techniques to optimize sequential access	LRU is not necessarily a good policy for sequential access files so sequential access files often take advantage of two special policies: <i>Free-behind</i> <i>Read-ahead</i>
Free-behind	removes a page from the buffer as soon as the next page is requested, with the assumption that we are now done with the old page and won't need it again for a long time.
Read-ahead	reads the requested page and several subsequent pages at the same time, with the assumption that those pages will be needed in the near future (caching)

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Recovery - Consistency Checking	(fsck in UNIX, chkdsk or scandisk in Windows) run at boot time or mount time. compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
Recovery - Log-Structured File Systems	Log-based transaction-oriented (a.k.a. journaling) filesystems borrow techniques developed for databases, guaranteeing that any given transaction either completes successfully or can be rolled back to a safe state before the transaction commenced
Recovery - Log-Structured File Systems implementation	<ul style="list-style-type: none">-All metadata changes are written sequentially to a log.-A set of changes for performing a specific task (e.g. moving a file) is a transaction.-As changes are written to the log they are said to be committed, allowing the system to return to its work.In the meantime, the changes from the log are carried out on the actual filesystem, and a pointer keeps track of which changes in the log have been completed and which have not yet been completed.-When all changes corresponding to a particular transaction have been completed, that transaction can be safely removed from the log.-At any given time, the log will contain information pertaining to uncompleted transactions only, e.g. actions that were committed but for which the entire transaction has not yet been completed.-- From the log, the remaining transactions can be completed,-- or if the transaction was aborted, then the partially completed changes can be undone.

UE18CS302_Unit 4_Revision_Class_Do_You_Know_That ?

Recovery - Sun's ZFS and Network Appliance's WAFL	No blocks of data are ever over-written in place. Rather the new data is written into fresh new blocks, and after the transaction is complete, the metadata (data block pointers) is updated to point to the new blocks. The old blocks can then be freed up for future use. Alternatively, if the old blocks and old metadata are saved, then a snapshot of the system in its original state is preserved. This approach is taken by WAFL.
log-structured file system	if system crash on a log-structured file system happen, then all transactions in the log must be completed.
NFS protocol	Implemented as a set of remote procedure calls (RPCs).
NFS Path-name translation	involves the parsing of a path name such as /usr/local/dir1/file.txt into separate directory entries, or components: (1) usr, (2) local, and (3) dir1
NFS cache	There are two caches: the file-attribute (inode-information) cache and the file-blocks cache . When a file is opened, the kernel checks with the remote server to determine whether to fetch or revalidate the cached attributes. The cached file blocks are used only if the corresponding cached attributes are up to date. The attribute cache is updated whenever new attributes arrive from the server. Cached attributes are, by default, discarded after 60 seconds
write-anywhere file layout (WAFL)	- is a powerful, elegant file system optimized for random writes

What are some physical characteristics of modern magnetic disk drives?	<p>A collection of two surface platters.</p> <p>One read/write head for each surface.</p> <p>Head assembly can be moved radially to one of a number of cylinders.</p> <p>Intersection of a cylinder and a surface is a track.</p> <p>Each track is logically divided into sectors.</p>
What is Seek Time?	The time taken to move heads to the proper cylinder.
What is Rotational Latency?	The time taken for desired sector to rotate into position under the head.
What is Transfer Time?	The time taken to transfer data in to main memory, once sector has been located.
What does it mean to achieve minimal variance?	Probability that time to process a particular request will deviate significantly from the average.
How does SCAN (Elevator Algorithm) work?	<p>Head moves as far as possible in one direction before reversing.</p> <p>Requests are serviced as head passes through their cylinders.</p> 
How does C-SCAN differ from SCAN?	Instead of reversing, the head jumps back to the beginning.

How does C-LOOK differ from C-SCAN?	Only moves head until no more requests in current direction.
When is Rotational Optimization optimal?	It can reduce rotational latency for high loads.
Why is Rotational Optimization not used for modern disks?	Modern disks have large "track buffers" which remove the need for this optimization, and using it could even make things worse.
What are some other useful optimizations?	<p>Overlapped seek - Arrange for multiple seek operations to go on in parallel.</p> <p>Sector interleaving - Avoid having logically consecutive sector numbers stored in physically consecutive sectors in a track.</p> <p>Read-ahead - When one sector is requested, read additional logically succeeding sectors in the same operation.</p> <p>Buffer cache - Dedicate a portion of main memory as a cache for recently used disk sectors.</p>
What does Overlapped seek require?	Controller support.
What does Read-ahead improve?	Sequential read access.
What does Sector Interleaving allow?	Allows slow CPUs to avoid "missed sectors"

For the other relevant Unit 4 concepts refer to the lecture supplements and relevant videos on PESU Academy



THANK YOU

Nitin V Pujari
Faculty, Computer Science
Dean - IQAC, PES University

nitin.pujari@pes.edu

For Course Deliverables by the Anchor Faculty click on www.pesuacademy.com