Assi_Uni

xChap 4

1) **What is disk inode? What is incore inode and what are the fields of both?**
   - inode is a data structure that describe a file or a folder.
   - disk indoe refer to the inode data structure as it is stored on the secondary memory and it is a static entity.
   - disk inode refers to the inode stored in disk within the inode list.
   - Filed of disk inode
     1. File owner identifier
     2. File type
     3. File access permission
     4. File accesses time
     5. Number of links to the file
     6. File size
     7. Disk address
   - In-core indoe refers to inode which is present in the main memory.
   - The kernel used it whenever a process wants to manipulate a file in the secondary memory.
   - It is a dynamic entity
   - Filed of incore inode contain following fields in addition to the fields of the disk inode
     8. The status of the in-core inode indicating whether the inode is locked
     9. Logical device number
     10. Inode number
     11. Pointer to other in-core inodes
     12. Reference count

2) **Why does incore inode differ from disk inode?**
   - Both inodes are two different representation of the same inode data structure
   - Disk inode stored on the secondary memory, Incore inode refer disk inode and stored in RAM
   - Disk inode is a static entity and the Incore inode is a dynamic entity
   - Incore inode contain some few extra fields than disk inodes that are only needed while the file is opened.
   - Incore inode file operation can be faster and more efficient because the operating system doesn’t have to constantly read from and write to the disk

3) **What is reference count? Why it is to be maintained?**
   - Reference count is the field of incore inode
   - Reference count indicating the number of instances of the file that are active
   - Reference count is to be maintained because without reference count difficult to allows the file system to manage storage efficiently
   - Difficult to determine which file and directory are still in use and which can be safely removed

- When reference count is reaches zero then only file or directory can removed
- Reference count help the file system to ensure that it does not waste storage space by keeping unnecessary files and directory

4) **What do you mean by inode is active?**
   - Inode is a data structure
   - Used to store information about file or directory
   - An inode is active when a process allocates it such as a opening a file
   - Inode is a active means that node is use by a process to do work on that like read or write so in this process that node consider as a active inode

5) **"If reference count of an incore inode is greater than 0,it can be put on free list of inode". Comment**
   - Reference count indicating the number of instances of the file that are active
   - Inode can be put on free list only if its reference count is 0
   - Meaning that the kernel can reallocate the in-core inode to another disk inode
   - Free list of inodes thus serves as a cache of inactive inodes
   - Kennel places the inode on the free list of inodes effectively caching the inode in case it is needed again soon
   - So if reference count of an incore inode is greater than 0 it can not be put on free list of inode

6) **What is the purpose of iget algorithm?**
   - Kernel identifies particular inodes by their file system and inode number and allocates in-core inode at the request of higher level algorithms
   - The algorithm iget allocates an in-core copy of an inode
   - The kernel maps the device number and inode number into a hash queue and searches the queue for the inode
   - It allocate one from the free list and locks it
   - The kernel than prepares to read the disk copy of the newly accessed inode into the in-core copy
   - It already knows the inode number and logical disk block that contain the inode according the how many disk inodes fit into a disk block

7) **Give the formula for calculating block number and to compute the byte offset.**
   - Block num = (inode number -1 / number of inodes per block) + start block of inode list
   - Compute the byte offset
     ((inode number -1) modulo (number of inodes per block)) * size of disk inode

8) **When an inode is locked?**
   - Reference count indicating the number of instances of the file that are active
   - When the reference count is greater than zero on that time inode is locked so that another process cannot allocated it
   - The kernel release the lock at the conclusion of the system call an inode is never locked across system calls

9) **How does kernel manipulate the system call?**
   - Kernel is a core component of an operating system
   - Kernel manage the system call
   - Application run in an area of memory knows as user space
   - System call connects to the operating systems kernel which executes in kernel space
   - When a application creates a system call it must first obtain the permission from the kernel
   - It achieve this using an interrupt request which pauses the current process and transfer control to the kernel
   - If request is permitted the kernel perform the request action like creating or deleting a file
   - As input the application receives the kernel output application resume the procedure after the input is received
   - When operation finished the kernel returns the result to application and then move data from kernel space to user space in memory

10) **When does kernel releases a lock?**
   - Reference count indicating the number of instances of the file that are active
   - When the reference count is greater than zero on that time inode is locked so that another process cannot allocated it
   - The kernel release the lock at the conclusion of the system call
   - Lock is set during the execution of a system call to prevent other process from accessing the inode while it is in use

11) **The lock is freed between system call to allow processes to share simultaneous access to file andreference count remains set between or across the system call. Why?**
   - Kernel manipulate the inode lock and reference count independently
   - Lock is set during the execution of a system call to prevent other process from accessing the inode while it is in use
   - Kernel releases the lock at the conclusion of the system call
   - Inode is never locked across system call
   - Kernel increment the reference count for every active reference to a file and decrement the reference count for every inactive reference to a file
   - Reference count thus remains set across multiple system calls
   - Lock is free between system call to allow process to share simultaneously accesses to

a file

- Reference count remain set between system call to prevent the kernel from relocating an active in-core inode
- Thus kernel can lock and unlock an allocated inode independent of the value of the reference count

12) Why kernel reports error if free list of inode is empty? And why not if free list of buffer is empty.

- Kernel reports an error when the free list of inodes is empty because inodes are used to represent file and directories in a file system
- If the free list of inode is empty it means that there are no more available inodes to represent new files or directory which could cause problems if a process try to create new file or directory
- Free list of buffer is used to track available memory for storing data temporary such as when reading or writing files
- If the free list of buffers is empty it simply means that there is no available memory for temporary storage but this can resolved by freeing up some memory or allocating more memory
- It does not cause a critical error that would prevent the system from functioning properly

13) Why the size of a file is limited to 4GB even if it can be greater than 16GB?

- The maximum size of a file in a file system is determine by the way that the file system stores data.
- In many file system maximum size of file is limited to 4GB because of the way that the file system uses a 32-bit numbers to address the data within the file
- This means that maximum size of file is limited to $2^{\wedge 32}$ bytes(4GB)
- However there are file system use a 64 bit number to address data within a file which allows for much large file sizes

14) What do you mean by read write and execute permission on directory?

- In a file system in an operating system each file and directory has a set of permission associated with it which determine what action can be performed on the file or directory
- These permission include read, write and execution permission
- The kernel store data for directory just as it stores data for an ordinary file using the inode structure and levels of direct and indirect blocks
- Process may read directories in the same way they read regular file but the kernel reserves exclusive rights to right a director thus insuring its correct structure
- Accesses permission of a directory have a following meaning: read permission on a directory allows a process to read directory
- Write permission allows a process to create new directory entries or remove old ones
- Execute permission allows a process to search the directory for file name

15) **For reading working inode directory why repeated use of algorithm bmap, bread, brelease areneeded?**

- bmap is a used in operating system locates the block on the disk that are used by the directory so that it can read the data and present it to the user in a readable format
- bread is used to locate the inode of a file in the file system directory
- brelease function is used to release the memory used by the inode data
- therefore bmap, bread and brelease are used to efficiently mange the retrieval and caching of inode data allowing it to provide fast and reliable access to directory contents

16) **What is the significance of lock fields for free block and free inode list of super block?**

- important aspect of file system design is the management of free blocks and free inodes, which are essential for allocating new files and directories
- superblock is a special data structure that contains metadata about the file system, including information about free blocks and free inodes
- some file systems, the superblock has lock fields associated with the free block and free inode lists
- These lock fields are used to ensure that multiple processes do not attempt to modify the free block and free inode lists simultaneously, which can result in data corruption or other errors
- When a process wants to allocate a new block or inode, it must first acquire the appropriate lock on the free block or free inode list in the superblock
- This ensures that no other process can allocate the same block or inode at the same time.
- Once the process has the lock, it can modify the free block or inode list as needed, allocate a new block or inode, and release the lock
- Locking the free block and free inode lists in the superblock helps to prevent data corruption and other errors that can occur when multiple processes attempt to allocate storage resources simultaneously

17) **How to identify an inode is free?**

- Inode is a data structure used to store information about a file or directory on a file system.
- Each file or directory on the file system has a corresponding inode that contains metadata about the file, such as ownership, permissions, and timestamps
- The kernel can determine whether an inode is free by inspection: If the file type field is clear, the inode is free.
- The kernel needs no other mechanism to describe free inodes. However, it cannot determine whether a block is free just by looking at it
- To identify if an inode is free, you need to check the inode bitmap.
- The inode bitmap is a data structure that keeps track of which inodes are currently in use and which are free.
- Each bit in the inode bitmap corresponds to an inode on the file system. If the bit is set to 0, the inode is free; if the bit is set to 1, the inode is in use
- If the reference count is zero or the inode is in the free list mean inode is free

18) Why list of free inode and free blocks are maintained in super block? Why not to search freeinodes and free data block directly from disk instead of maintaining in super block.

- the superblock is a key data structure that stores information about the file system, including the total number of blocks and inodes, as well as the location of important data structures such as the inode table and the free block and inode bitmaps
- One of the reasons why free inodes and free data blocks are maintained in the superblock is to improve performance.
- If the file system had to search the entire disk for free inodes and data blocks every time a file was created or modified, this would be a slow and inefficient process.
- maintaining lists of free inodes and data blocks in the superblock, the file system can quickly locate and allocate these resources when needed
- Another reason to prevent fragmentation.
- When files are created and deleted on a file system, the free space on the disk becomes fragmented, meaning that it is scattered throughout the disk in small pieces.
- By keeping track of free blocks and inodes in the superblock, the file system can more effectively allocate space for new files and minimize fragmentation

19) Give steps for placing inode in free list when super block free inode list is empty.

- Identify a block of free disk space that can be used to store the new inode
- Create a new inode in the chosen block and initialize its values, including the file type, permissions, owner, group, timestamps, and pointers to data blocks
- Mark the new inode as free by setting its status field to "free"
- Locate the head of the free inode list in the superblock
- Update the pointer in the new inode to point to the inode currently at the head of the free inode list
- Update the pointer in the superblock to point to the new inode, making it the new head of the free inode list
- Update the superblock to reflect the change in the number of free inodes
- Write the updated superblock and new inode to disk
- following these steps, the new inode will be successfully added to the free list, allowing it to be used to store a new file or directory in the file system

20) What is the use of remembered inode? What are the advantages?

- An inode, short for "index node," is a data structure used by file systems to store information about a file or directory on a disk.
- The inode contains metadata about the file, such as its owner, permissions, size, and timestamps, as well as pointers to the blocks on the disk where the file's contents are stored
- A "remembered inode" is a term that typically refers to an inode that is kept in memory by the operating system, rather than being read from disk each time the file is accessed.
- By keeping frequently accessed inodes in memory, the operating system can reduce the amount of time it takes to access the file, since it doesn't need to go to the disk to look up the inode information

- Advantages
  1. **Improved performance :** By keeping frequently accessed inodes in memory, the operating system can reduce the amount of time it takes to access files, which can lead to improved overall system performance
  2. **Reduced disk I/O :** By minimizing the number of times the operating system needs to read inode information from disk, the use of remembered inodes can reduce disk I/O, which can help to extend the lifespan of disk drive
  3. **Reduced system overhead :** By keeping inode information in memory, the operating system can reduce the amount of system overhead required for disk access, which can help to improve system stability and reliability

21) Explain race condition arises in assigning inode. What action does kernel take to avoid racecondition?

- inode is a data structure that stores metadata about a file, such as ownership, permissions, timestamps, and disk block locations
- When a new file is created, the kernel must assign a unique inode number to the file, which it does by incrementing a counter stored in a superblock on the disk
- A race condition can arise when multiple processes or threads try to create files simultaneously, and they all try to access and update the inode counter at the same time.
- If two or more processes read the same value of the counter, they will all assign the same inode number to their respective files, leading to conflicts and inconsistencies in the filesystem
- To avoid this race condition, the kernel employs a technique called serialization or mutual exclusion, where it ensures that only one process can access and modify the inode counter at a time.
- This is usually achieved using locks which prevent other processes from accessing the critical section of code until the current process has finished its task
- Atomic operations are indivisible and non-interruptible, so even if multiple processes try to access the counter simultaneously, they will execute the operation one after another, ensuring that each file gets a unique inode number

22) Why processes freeing an inode checks whether the super block is locked? What would happenif it is not locked?

- When a process is freeing an inode, it typically needs to update the corresponding information in the superblock, which is a data structure that contains information about the filesystem, such as the total number of inodes and blocks, and free inode and block lists
- If the superblock is not locked when the process attempts to free the inode, it can lead to a race condition where multiple processes could simultaneously modify the

superblock, potentially corrupting the data structure and causing data loss or filesystem corruption

- Locking the superblock ensures that only one process at a time can modify it, preventing concurrent modifications and ensuring the consistency of the filesystem
- checking whether the superblock is locked before freeing an inode is a necessary precaution to prevent data loss or filesystem corruption that could occur as a result of concurrent modifications to the superblock by multiple processes

23) If super block is not locked then discuss the race condition arises in reading a new set of free inodes?

- superblock contains important metadata about the file system, such as the number of blocks, the size of each block, and the location of the root directory.
- In many operating systems, a pool of free inodes is maintained in the superblock.
- When a new file needs to be created, an inode is allocated from this pool of free inodes
- A race condition can arise when multiple processes or threads attempt to allocate an inode from the pool of free inodes at the same time, and the superblock is not locked.
- If the superblock is not locked, two or more processes could end up reading the same set of free inodes and allocate the same inode to different files, leading to data corruption and other issues
- To prevent this race condition, the superblock should be locked while the free inode list is being read and updated.
- This ensures that only one process can access the free inode list at a time and prevents multiple processes from allocating the same inode.
- Alternatively, the file system could use a more sophisticated allocation algorithm that avoids the need for a shared pool of free inodes

24) Why free block are maintained as linked list and why not in super block?

- free blocks refer to the blocks on the storage device that are not currently being used by any file or directory.
- These blocks need to be tracked and managed by the operating system in order to allocate them to files or directories as needed
- One way to maintain a list of free blocks is to include it in the super block, which is a data structure that contains information about the file system as a whole
- Accessing the super block for every request for a free block can be slow and can cause performance issues, especially for large file systems
- Updating the super block every time a block is allocated or freed can lead to a high degree of contention and synchronization issues, which can affect system performance
- To avoid these issues, many file systems maintain a separate linked list of free blocks.
- This list is typically stored in a fixed location on the disk and is updated whenever a block is allocated or freed.
- By keeping this list separate from the super block, the operating system can quickly allocate and free blocks without having to access the super block every time.
- This can improve performance and reduce contention issues

25) **Is it necessary to maintain incore inode? What would happen if it is not maintained?**

- It is necessary to maintain the in-core inode because it is used by the operating system to access and manipulate files.
- When a file is opened, the operating system reads the inode from disk into memory and creates an in-core inode.
- The in-core inode is then used to track changes to the file, such as when it is read, written, or deleted
- If the in-core inode is not maintained in advance by the operating system, it can lead to various issues.
- For example, the operating system may not be able to access or manipulate files correctly, leading to data corruption, file system errors, or even system crashes.
- if the in-core inode is not updated correctly, it can result in inconsistencies between the in-core inode and the on-disk inode, which can cause data loss
- Therefore, it is important to maintain the in-core inode in advance operating systems to ensure the proper functioning of the file system and prevent potential data loss or system crashes