Comprehensive Revision Notes on Backup and Inode Concepts

Introduction to Backup Strategies

Types of Backups

In the session, three main types of backups were discussed:

1. **Full Backup**: Captures all data in the given directories. Typically scheduled on Sundays for a complete dataset capture【4:0†source】.
2. **Incremental Backup**: Only files that have changed since the last backup are included. This type reduces data redundancy and saves space【4:0†source】.
3. **Differential Backup**: Includes all files that have changed since the last full backup. Offers a balance between full and incremental backups【4:0†source】.

Backup Implementation

- **Backup Functionality**: A single `tar-XVCZ` function is used regardless of backup type, with variations only in file selection【4:0†source】.
- **File Markers**: After a backup is executed, a marker file (e.g., `.full_backup.txt`) is created. This serves as a reference for future backups to identify modified or newly created files【4:0†source】.

File Selection for Backup

- File selection is based on using the `find` command with the `-newer` option. This command locates files newer than the marker file【4:0†source】.

RSync for Backup Synchronization

compression)【4:4†source】.

- **Mirroring**: Ensures that the source matches the destination exactly by adding the `--delete` option【4:12†source】.
- **Bandwidth Limiting**: Implemented using `--bwlimit` to prevent network resource exhaustion【4:12†source】.

Understanding Inodes in Linux File Systems

Inode Structure

- **Definition**: An inode is a data structure on a filesystem that stores information about a file except its name or its actual data【4:1†source】.
- **Attributes Stored in Inodes**: File size, device ID, user ID (UID), group ID (GID), and permission data【4:1†source】.
- **Linking**: Inodes are linked to files by mapping multiple filenames to a single inode in the case of hard links【4:1†source】【4:8†source】.

Inode Mapping and File Structure

- **Inode to File Mapping**: One-to-one mapping between a file and an inode. However, one inode can have multiple hard links (filenames) pointing to it【4:5†source】.
- **Inode to Data Block Mapping**: Inodes point to data blocks on the disk, illustrating a one-to-many relationship where one inode can associate with multiple data blocks【4:5†source】.
- **Usage Considerations**: File modifications (e.g., using `chmod`) affect the metadata stored in inodes, not the file data itself【4:15†source】.

Key Concepts in Inode Utilization

- **Hard Link Concept**: Enables multiple filenames to reference a single inode. Useful for saving storage and ensuring data redundancy【4:15†source】.
- **Inode Limitation**: Inodes are pre-allocated during filesystem creation and are finite, which can constrain the total number of files on a filesystem【4:1†source】【4:17†source】.

Checksum Generation

- **Purpose**: Validate the integrity of backups and ensure no data corruption has occurred【4:3†source】.
- **Implementation**: Use `sha256sum` to generate and validate checksums before and after backup operations【4:18†source】.

Summary

This session effectively covered the essentials of backup strategies and inode management in Linux systems. Understanding these concepts is vital for efficient data management, ensuring data integrity, and performing recovery operations in a system administration context.