**Chapter 1: Overview of Disk Devices in Linux**

**1.1 Introduction**

This chapter provides a basic understanding of disk devices, their types, interfaces, and essential terminology in Linux systems.

**1.2 Types of Storage Devices**

* **Hard Disk Drives (HDDs)**: Traditional storage with spinning magnetic disks; cost-effective but slower.
* **Solid State Drives (SSDs)**: Faster storage using flash memory; more durable and efficient.
* **NVMe Drives**: High-speed SSDs using the PCIe interface for superior performance.

**1.3 Disk Interfaces**

* **SATA (Serial ATA)**: Common interface for HDDs and SSDs, with versions offering different speeds (SATA I, II, III).
* **SCSI (Small Computer System Interface)**: Older standard used in enterprise environments; supports multiple devices.
* **NVMe (Non-Volatile Memory Express)**: Protocol for high-speed SSDs via PCIe, offering fast data transfer.
* **USB (Universal Serial Bus)**: Used for external storage; convenient but generally slower than internal interfaces.

**1.4 Understanding Disk Terminology**

* **Sectors**: Smallest unit of storage on a disk (usually 512 bytes or 4K bytes).
* **Tracks**: Concentric circles on a disk platter where data is recorded.
* **Cylinders**: Vertical alignment of tracks across multiple platters.

**Secure File Deletion Methods**

To ensure that a file is deleted and not recoverable, you need to overwrite it so that the original data cannot be reconstructed. Here are methods and tools to securely delete files and make them unrecoverable:

**1. Use the shred Command**

**shred** is a command-line utility that overwrites a file to make it difficult to recover.

**Command:**

shred -u -v -z /path/to/file

* **-u**: Remove the file after shredding.
* **-v**: Verbose mode, which shows the progress.
* **-z**: Add a final pass with zeros to hide the shredding process.

**Example:**

shred -u -v -z /home/user/secretfile.txt

**2. Use the wipe Command**

**wipe** is another utility for securely deleting files by overwriting them.

**Installation:**

sudo apt-get install wipe

**Command:**

wipe /path/to/file

**Example:**

wipe /home/user/secretfile.txt

**3. Use dd for Overwriting (Manual Method)**

**dd** can be used to overwrite the file with random data or zeros.

**Command:**

dd if=/dev/urandom of=/path/to/file bs=1M

* **if=/dev/urandom**: Input file (random data).
* **of=/path/to/file**: Output file (the file to overwrite).
* **bs=1M**: Block size (1 MB).

After overwriting, you should remove the file:

rm /path/to/file

**Example:**

dd if=/dev/urandom of=/home/user/secretfile.txt bs=1M

rm /home/user/secretfile.txt

**4. Use badblocks for Overwriting**

**badblocks** can be used to securely wipe a disk by writing patterns to it. While typically used for checking disk integrity, it can be employed for secure erasure.

**Command:**

sudo badblocks -w -o /path/to/output /dev/sdX

* **-w**: Write mode (overwrite).
* **-o**: Output file for logging bad blocks.
* **/dev/sdX**: Disk or partition to overwrite.

**Example:**

sudo badblocks -w -o /home/user/badblocks.log /dev/sda

**Note:** Using badblocks will overwrite the entire disk. Ensure you specify the correct disk and understand that this will erase all data.

**5. Use Secure Deletion Tools**

**secure-delete** package includes tools like sfill, smem, and srm for secure file deletion.

**Installation:**

sudo apt-get install secure-delete

**Command:**

* **srm** (secure remove):
* srm /path/to/file

**Example:**

srm /home/user/secretfile.txt

**6. Use File System-Level Tools**

For complete disk wiping, which is useful if you want to ensure that no file recovery is possible, use tools like dd or shred on the whole partition or disk:

**Command:**

dd if=/dev/zero of=/dev/sdX bs=1M

* **if=/dev/zero**: Input file (zeros).
* **of=/dev/sdX**: Disk or partition (overwrite all data).

**Example:**

dd if=/dev/zero of=/dev/sda bs=1M

**Summary**

* **shred**: Overwrites files multiple times and then deletes them.
* **wipe**: Securely deletes files by overwriting.
* **dd**: Can be used to overwrite files or disks with random data or zeros.
* **badblocks**: Writes patterns to disks, suitable for complete disk wiping.
* **secure-delete**: Provides additional secure deletion tools like srm.
* **File System-Level Tools**: For complete disk wiping, use dd or similar tools.

**Adding a Disk to a Running VM**

1. **Open VMware Player** and select the virtual machine to which you want to add a disk.
2. Go to **Edit virtual machine settings**.
3. Click **Add** and select **Hard Disk**.
4. Choose **SCSI** (recommended) or **IDE**, then click **Next**.
5. Choose the disk type (typically **Create a new virtual disk**), then click **Next**.
6. Specify the disk size and other settings, then click **Finish**.
7. Ensure the new disk is attached to the VM and start the VM.

**Scanning the New Disk**

1. **Open a terminal** in your Linux VM.
2. **Scan for new disks** using the following command:

echo "- - -" > /sys/class/scsi\_host/host0/scan

This command rescans the SCSI bus for new devices.

1. **Verify the new disk** using lsblk:

lsblk

**Formatting and Partitioning the Disk**

**Step 1: Create a Primary Partition**

1. **Run fdisk** to modify disk partitions:

sudo fdisk /dev/sda

Replace /dev/sda with the actual device name of your new disk.

1. **Create a new partition:**
   * Press n to create a new partition.
   * Press p for a primary partition.
   * Press 1 to specify the partition number.
   * Press Enter to accept the default starting sector.
   * Specify the desired partition size (e.g., +2G for 2 GB).
   * Press t to change the partition type.
   * Type 8e to set the type to Linux LVM.
   * Press w to write the changes and exit.
2. **Update the partition table:**

sudo partprobe

**Step 2: Create Additional Partitions (Optional)**

You can create more partitions as needed, following the same steps as above.

**Formatting the Partition**

1. **Choose a file system:** The choice of file system depends on your needs (e.g., Ext4, XFS, Btrfs).
2. **Format the partition:**

sudo mkfs.xfs /dev/sda1

Replace /dev/sda1 with the actual partition name.

**Mounting the Partition**

1. **Create a mount point:**

sudo mkdir /mnt/new\_disk

1. **Mount the partition:**

sudo mount /dev/sda1 /mnt/new\_disk

Replace /dev/sda1 and /mnt/new\_disk with the appropriate values.

**Verifying the Changes**

1. **Check the mounted partition:**

df

1. **Create and verify files:**

cd /mnt/new\_disk

touch test.txt

ls

**Attaching and Preparing the Disk**

1. **Checking Existing Disks (lsblk)**:
   * The lsblk command lists information about all available block devices. This includes hard drives, partitions, and any attached storage media. It provides details like size, type, mount points, and file system types.
2. **Attaching a New Disk to the VM**:
   * If you’re using VMware, you need to attach a new virtual disk to the virtual machine (VM) manually through VMware’s interface. Once done, you can scan for this newly attached disk without rebooting the VM using:
   * for x in `ls /sys/class/scsi\_host/`; do echo "- - -" > /sys/class/scsi\_host/$x/scan; done

This command loops through the SCSI host controllers and forces the kernel to rescan the bus for new devices.

1. **Partitioning the Disk (fdisk)**:
   * The fdisk utility is used for creating and modifying partitions on a disk. After running fdisk /dev/sda, you can create a primary partition by selecting n (for new partition) and p (for primary). Pressing Enter accepts default values, and w writes the partition table to the disk. Finally, partprobe is run to update the kernel with the new partition information without rebooting the system.

**Formatting the Partition**

1. **Formatting with EXT2 (mke2fs)**:
   * The mke2fs command creates an EXT2 file system on the partition. EXT2 is an older file system without journaling capabilities, meaning that it doesn’t keep track of changes that are about to be made. This makes it faster but riskier in the event of a crash or power failure.
   * Example:
   * mke2fs /dev/sda1
2. **Checking the Filesystem with blkid**:
   * blkid returns information about the block device, including its UUID and file system type.
   * blkid /dev/sda1
3. **Viewing Filesystem Features with tune2fs**:
   * tune2fs displays detailed information about the file system, including its features. You can check features such as whether it has a journal (EXT3/EXT4) or supports extended attributes (EXT4).
   * tune2fs -l /dev/sda1 | grep feature

**Converting EXT2 to EXT3**

1. **Why Convert to EXT3?**
   * EXT3 adds journaling, which keeps a log of changes to files and metadata, ensuring quick recovery after a crash. This feature makes the file system more robust compared to EXT2.
   * Conversion does not erase the data. Instead, it adds the necessary structures for journaling to the existing EXT2 partition.
2. **Command for Conversion**:
   * Use the tune2fs command with the -j flag to add a journal to the existing EXT2 partition, effectively converting it to EXT3:
   * tune2fs -j /dev/sda1
3. **Verify the Conversion**:
   * After conversion, you can verify the file system’s features again using:
   * blkid /dev/sda1
   * tune2fs -l /dev/sda1 | grep feature

**Converting EXT3 to EXT4**

1. **Why Convert to EXT4?**
   * EXT4 introduces several improvements over EXT3, such as:
     + **Extents**: Instead of tracking individual blocks, EXT4 tracks contiguous blocks, improving performance for large files.
     + **Faster fsck**: File system checks are faster due to the new structure.
     + **Larger volume support**: EXT4 can handle volumes up to 1 exabyte and files as large as 16 terabytes.
     + **Backward compatibility**: EXT4 is backward compatible with EXT3. You can still mount an EXT4 file system as EXT3 if you don’t use features like extents.
2. **Command for Conversion**:
   * You can convert an EXT3 file system to EXT4 by enabling the following features:
     + **Extents**: Improves the way large files are handled.
     + **Uninitialized Block Groups**: Improves fsck times by tracking unused blocks.
     + **Directory Indexing**: Speeds up directory lookups by indexing directory contents.
   * tune2fs -O extents,uninit\_bg,dir\_index /dev/sda1
3. **Checking the Conversion**:
   * Run blkid to confirm that the file system type has been updated to EXT4. After the conversion, you should also remount the file system to begin using the new EXT4 features:
   * mount /dev/sda1 /test

**Enabling and Disabling the Journaling Feature**

1. **Why Disable Journaling?**
   * Disabling journaling can improve write performance but increases the risk of data corruption in case of a system crash. This might be desired in certain scenarios like temporary storage or when performance is prioritized over data safety.
2. **Disabling Journaling**:
   * To disable journaling, you can remove the has\_journal feature:
   * umount /test
   * tune2fs -O ^has\_journal /dev/sda1
   * tune2fs -l /dev/sda1 | grep feature

The ^ symbol indicates removal of the journaling feature.

1. **Re-enabling Journaling**:
   * If you decide to re-enable journaling, use the following command:
   * tune2fs -O has\_journal /dev/sda1
   * tune2fs -l /dev/sda1 | grep feature

**Mounting the Partition Automatically with /etc/fstab**

1. **What is /etc/fstab?**
   * The /etc/fstab file contains information about where partitions should be mounted at boot time. By editing this file, you can set your newly created EXT4 partition to mount automatically on system startup.
2. **Adding the Partition to /etc/fstab**:
   * Edit the /etc/fstab file using a text editor:
   * vi /etc/fstab
   * Add an entry similar to:
   * /dev/sda1 /test ext4 defaults 0 2
     + /dev/sda1: The device to be mounted.
     + /test: The mount point.
     + ext4: The file system type.
     + defaults: Mount options (e.g., read-write, suid, dev, exec, auto).
     + 0: Dump flag (usually 0 to disable dump).
     + 2: The order in which fsck checks the file system on boot.
3. **Verifying the Setup**:
   * After editing /etc/fstab, test it by unmounting and remounting:
   * umount /test