**Module 17**

## Linux server - Configure local storage Assignment

* Level Basic to Advance
  1. What is…
  2. /dev/sda

The "/dev/sda" is a common naming convention for the first hard disk drive (HDD) in a Linux system. In Linux, devices are represented as files, and "/dev" is the directory that holds device files. The "sda" part indicates the first SCSI (Small Computer System Interface) disk in the system, which can be an HDD or a Solid-State Drive (SSD).

The naming convention for hard drives in Linux follows a pattern, where "sd" stands for SCSI disk, and "a" represents the first disk. If there are additional disks, they are named "sdb," "sdc," and so on.

It's important to note that the naming convention may vary depending on the Linux distribution or the specific hardware configuration of the system. For example, if the system uses an IDE or SATA interface, the naming convention might be different (e.g., "/dev/hda" for IDE or "/dev/sda" for SATA).

In summary, "/dev/sda" refers to the first SCSI disk in a Linux system, typically representing the primary hard disk drive.

* 1. /dev/sda1

The "/dev/sdb" is a naming convention for the second hard disk drive (HDD) or Solid-State Drive (SSD) in a Linux system. Similar to "/dev/sda", "/dev/sdb" is also a device file located in the "/dev" directory that represents a storage device.

In Linux, the naming convention for hard drives follows a pattern where "sd" denotes SCSI disk, and the letter following it represents the order of the disk. So, "/dev/sdb" signifies the second SCSI disk in the system.

It's important to note that the naming convention may differ based on the Linux distribution or the hardware configuration of the system. For instance, if the system uses an IDE or SATA interface, the naming convention might be different (e.g., "/dev/hdb" for IDE or "/dev/sdb" for SATA).

In summary, "/dev/sdb" refers to the second SCSI disk in a Linux system, typically representing an additional hard disk drive or SSD.

* 1. /dev/sda2

The "/dev/sda2" is a specific partition on the first hard disk drive (HDD) or Solid-State Drive (SSD) in a Linux system. In Linux, partitions are created on storage devices to divide them into separate logical sections.

The naming convention for partitions follows a pattern where "sd" represents SCSI disk, "a" denotes the order of the disk, and the number following it represents the specific partition on that disk. So, "/dev/sda2" signifies the second partition on the first SCSI disk in the system.

The numbering of partitions typically starts from "1" for the first partition on a disk. However, the number "1" is commonly reserved for the primary partition, and subsequent partitions are numbered starting from "2" and onwards.

It's important to note that the naming convention may vary depending on the Linux distribution or the specific partitioning scheme used. For example, if the disk is partitioned using the GUID Partition Table (GPT) scheme, the partition might be named as "/dev/sda2" or "/dev/nvme0n1p2" for NVMe drives.

In summary, "/dev/sda2" represents the second partition on the first SCSI disk in a Linux system, providing a specific logical section of the storage device for data organization and management.

* 1. /dev/vda

The "/dev/vda" is a naming convention for a virtual disk drive in a virtualized environment, typically used by virtualization technologies such as KVM (Kernel-based Virtual Machine) or QEMU.

In virtualized environments, virtual disk drives are represented as device files, and "/dev/vda" is a common naming convention for the first virtual disk drive in the system.

The "vda" part of the name does not follow the SCSI disk naming convention used in physical systems. Instead, it is a representation specific to virtualization. The letter "v" stands for virtual, and "da" indicates the first virtual disk drive.

It's worth noting that the naming convention for virtual disk drives can vary depending on the virtualization technology and configuration. For example, in some virtualization platforms, the virtual disk drives may be named as "/dev/sda" or "/dev/xvda".

In summary, "/dev/vda" refers to the first virtual disk drive in a virtualized environment, typically associated with virtualization technologies like KVM or QEMU.

* 1. /dev/vda1

The "/dev/vda1" is a specific partition on the first virtual disk drive in a virtualized environment. In virtualization technologies like KVM or QEMU, virtual disk drives are represented as device files, and partitions can be created on these virtual disks.

The naming convention for virtual disk drive partitions follows a pattern where "vda" represents the virtual disk drive, and the number following it represents the specific partition on that virtual disk drive. So, "/dev/vda1" signifies the first partition on the first virtual disk drive in the system.

The numbering of partitions on virtual disk drives typically starts from "1" for the first partition. However, the naming convention may differ depending on the virtualization technology and configuration.

It's important to note that the naming convention for virtual disk drive partitions may vary across different virtualization platforms. Some platforms may use different naming schemes such as "/dev/sda1" or "/dev/xvda1".

In summary, "/dev/vda1" represents the first partition on the first virtual disk drive in a virtualized environment, providing a specific logical section of the virtual disk drive for data organization and management.

* 1. What is the use of “df “command?

The "df" command is a command-line utility in Unix-like operating systems (including Linux) that is used to display information about the available disk space on file systems. "df" stands for "disk free."

When you run the "df" command without any arguments, it provides a summary of disk space usage for all mounted file systems on the system. It displays information such as the file system's device, total size, used space, available space, percentage of space used, and the mount point.

The "df" command is particularly useful for monitoring disk space usage, checking if a file system is running out of space, and identifying which directories or partitions are consuming the most disk space. It helps you make informed decisions about disk usage, identify potential storage issues, and take appropriate actions such as cleaning up unnecessary files or resizing partitions.

Here's an example output of the "df" command:

bash

Copy code

Filesystem 1K-blocks Used Available Use% Mounted on

/dev/sda1 10239872 3819024 6428556 38% /

tmpfs 2048000 124 2047876 1% /dev/shm

/dev/sdb1 52428800 1534564 48676888 4% /data

In the above example, the output displays information about three file systems: "/dev/sda1", "/", and "/dev/sdb1". It provides details about their total size, used space, available space, percentage of space used, and their respective mount points.

Overall, the "df" command is a useful tool for checking disk space usage and managing storage resources in a Unix-like operating system.

* 1. From which command we get UUID of file system?

The "blkid" command is commonly used to retrieve the UUID (Universally Unique Identifier) of a file system in Linux. The "blkid" command is a utility that examines devices and provides information about their attributes, including the UUID.

To obtain the UUID of a file system using the "blkid" command, follow these steps:

Open a terminal or command prompt.

Execute the following command:

Copy code

blkid

This command will display information about all the available block devices and their associated file systems.

Look for the line that corresponds to the file system whose UUID you want to retrieve. The line will contain the file system's device path (e.g., "/dev/sda1") and its associated UUID.

For example, the output may look like this:

bash

Copy code

/dev/sda1: UUID="3ec012c1-9fb8-4c68-a11a-9eabc1234567" TYPE="ext4" PARTUUID="12345678-01"

In this case, the UUID of the file system "/dev/sda1" is "3ec012c1-9fb8-4c68-a11a-9eabc1234567".

The UUID is a unique identifier assigned to a file system, and it can be useful for various purposes such as mounting file systems using UUID instead of device names, configuring entries in the "/etc/fstab" file, or referencing specific file systems in configuration files.

Note that the "blkid" command typically requires root (superuser) privileges to access and read device information, so you may need to prefix it with "sudo" or run it as the root user.

* 1. I want to use /dev/sdb1, which command will used? (wright down full argument)

To use the "/dev/sdb1" partition, you can use the "mount" command in Linux. The "mount" command is used to mount file systems, including partitions, onto specified mount points in the directory tree.

To mount the "/dev/sdb1" partition, you can execute the following command:

bash

Copy code

sudo mount /dev/sdb1 <mount\_point>

Replace "<mount\_point>" with the actual directory path where you want to mount the partition. For example, if you want to mount it at "/mnt/data", the command would be:

bash

Copy code

sudo mount /dev/sdb1 /mnt/data

By default, the "mount" command assumes the file system type based on the partition's file system type. If you need to specify the file system type explicitly, you can use the "-t" option followed by the file system type. For example, if it's an ext4 file system, the command would be:

bash

Copy code

sudo mount -t ext4 /dev/sdb1 /mnt/data

Remember to replace "/mnt/data" with the actual mount point directory of your choice.

After executing the mount command, the "/dev/sdb1" partition will be mounted at the specified mount point, and you can access its contents through that directory.

* 1. Where we find all hardware info?

In Linux, you can find hardware information from various sources. Here are some common methods to retrieve hardware information:

/proc directory: The "/proc" directory contains a virtual file system that provides information about the running Linux kernel and system hardware. You can explore the "/proc" directory and its subdirectories to find detailed information about CPU, memory, disks, devices, and more. For example, you can view the CPU information in "/proc/cpuinfo" or check disk information in "/proc/partitions".

lspci: The "lspci" command displays information about the PCI buses and devices connected to them. It provides details about hardware devices such as graphics cards, network adapters, and sound cards. Running the "lspci" command in the terminal will output a list of hardware devices along with their vendor and device IDs.

lsusb: The "lsusb" command displays information about USB buses and the devices connected to them. It shows details about USB devices, including their vendor and product IDs, which can be helpful in identifying specific USB devices.

lshw: The "lshw" command (short for "list hardware") provides detailed information about the hardware configuration of the system. It offers a comprehensive view of hardware components, including CPUs, memory, disks, network interfaces, and more. Running "sudo lshw" in the terminal will generate an extensive report of the system's hardware information.

dmidecode: The "dmidecode" command provides information about the system's hardware as reported by the system BIOS or UEFI firmware. It can display details about the motherboard, BIOS version, CPU, memory, and other hardware components. Running "sudo dmidecode" in the terminal will display the hardware information gathered from the system's firmware.

hwinfo: The "hwinfo" command is a comprehensive hardware information tool that provides detailed information about various hardware components. It offers extensive details about CPUs, memory, disks, network interfaces, and more. Running "sudo hwinfo" in the terminal will generate a report with comprehensive hardware information.

These methods can be used individually or in combination to gather hardware information in Linux. Note that some of these commands may require root (superuser) privileges to access certain system files and retrieve detailed hardware information.

* 1. Which command is used to create MBR partition?

To create a Master Boot Record (MBR) partition in Linux, you can use the "fdisk" command. "fdisk" is a command-line utility used for managing disk partitions on Linux systems.

Here's a step-by-step guide on how to create an MBR partition using the "fdisk" command:

Open a terminal or command prompt.

Run the following command with the disk device you want to partition (e.g., "/dev/sdb"):

bash

Copy code

sudo fdisk /dev/sdb

Replace "/dev/sdb" with the actual device name of the disk you want to partition. Make sure you have the appropriate permissions (superuser or sudo) to run the command.

After running the command, you will enter the "fdisk" command-line interface. It will display information about the disk you specified.

Use the following commands in the "fdisk" interface to create an MBR partition:

Press "n" to create a new partition.

Choose the primary partition type by pressing "p".

Specify the partition number (e.g., "1" for the first partition).

Specify the starting sector of the partition (e.g., press Enter for the default value to use the entire disk).

Specify the size of the partition (e.g., "+2G" for a 2 GB partition).

Press "w" to write the partition table changes and exit.

After writing the partition table changes, you can format the newly created partition with a file system of your choice using the appropriate formatting command (e.g., "mkfs.ext4" for ext4 file system).

Please note that creating or modifying partitions can be a sensitive operation that affects the data on the disk. It's essential to exercise caution and double-check the disk and partition information before executing any partitioning commands.

* 1. Which command is used to create GPT partition?

To create a GUID Partition Table (GPT) partition in Linux, you can use the "gdisk" command. "gdisk" is a specialized command-line utility for managing GPT partitions on Linux systems.

Here's a step-by-step guide on how to create a GPT partition using the "gdisk" command:

Open a terminal or command prompt.

Run the following command with the disk device you want to partition (e.g., "/dev/sdb"):

bash

Copy code

sudo gdisk /dev/sdb

Replace "/dev/sdb" with the actual device name of the disk you want to partition. Ensure you have the appropriate permissions (superuser or sudo) to run the command.

After executing the command, you will enter the "gdisk" command-line interface. It will display information about the disk you specified.

Use the following commands in the "gdisk" interface to create a GPT partition:

Press "n" to create a new partition.

Specify the partition number (e.g., "1" for the first partition).

Specify the starting sector of the partition (e.g., press Enter for the default value to use the next available sector).

Specify the ending sector of the partition (e.g., press Enter for the default value to use the remaining space on the disk).

Specify the partition type code (e.g., press Enter for the default value to create a Linux data partition).

Optionally, provide a name for the partition (you can leave it blank by pressing Enter).

Press "w" to write the partition table changes and exit.

After writing the partition table changes, you can format the newly created partition with a file system of your choice using the appropriate formatting command (e.g., "mkfs.ext4" for ext4 file system).

Please note that creating or modifying partitions can be a sensitive operation that affects the data on the disk. It's essential to exercise caution and double-check the disk and partition information before executing any partitioning commands. Additionally, using GPT partitions requires support from the system's BIOS or UEFI firmware.

* 1. What is sda, sdb, sdc, sdb1, sdb2, sdb3?

The terms "sda," "sdb," "sdc," "sdb1," "sdb2," and "sdb3" are related to the naming conventions of storage devices and their partitions in Linux systems. Let's break down each term:

sda: "sda" refers to the first SCSI (Small Computer System Interface) disk in the system. In modern Linux systems, this term is commonly used to represent the first hard disk drive (HDD) or solid-state drive (SSD).

sdb: "sdb" represents the second SCSI disk in the system. It refers to the second HDD or SSD connected to the system.

sdc: "sdc" signifies the third SCSI disk in the system. It represents the third HDD or SSD connected to the system.

sdb1: "sdb1" represents the first partition on the second SCSI disk (sdb). When a disk is divided into multiple logical sections called partitions, the numbering starts from "1." Therefore, "sdb1" indicates the first partition on the second disk.

sdb2: "sdb2" refers to the second partition on the second SCSI disk (sdb). If the disk is further divided into multiple partitions, subsequent partitions are numbered accordingly.

sdb3: "sdb3" represents the third partition on the second SCSI disk (sdb). Similarly, if there are more partitions on the disk, they would be numbered accordingly.

The naming convention follows a pattern where "sd" stands for SCSI disk, followed by a letter denoting the order of the disk (e.g., "a" for the first disk, "b" for the second disk), and a number indicating the partition number (if applicable).

It's important to note that while "sda," "sdb," etc., are commonly used in Linux systems, the naming conventions may vary depending on the specific configuration, type of storage devices (such as NVMe drives), or the partitioning scheme used (MBR or GPT).

* 1. What is the use of swap partition?

A swap partition is a dedicated area on a storage device (usually a hard disk drive or solid-state drive) that serves as virtual memory in a Linux or Unix-like operating system. Its primary purpose is to extend the available memory (RAM) by providing additional space for storing data when the physical memory is full or when the system needs to swap out idle or less frequently used data.

Here are some key uses and benefits of a swap partition:

Memory Expansion: When the system's physical memory (RAM) is fully utilized, the swap partition provides additional memory capacity. It allows the system to continue running by temporarily moving less frequently used data from RAM to the swap space, freeing up RAM for more active processes.

Memory Management: The swap partition acts as a form of memory management. It allows the operating system to optimize memory usage by swapping out inactive processes or pages to the swap space, making room for more important or active processes in the RAM.

Hibernate and Sleep Modes: When a system enters hibernate or sleep mode, the contents of the RAM need to be saved to a persistent storage medium. A swap partition is used for storing the hibernation image or suspend-to-disk file, allowing the system to resume from hibernation or sleep state later.

System Stability: Having a swap partition can enhance system stability by preventing out-of-memory errors. If the system runs out of physical memory, it can still continue functioning by utilizing the swap space to store less frequently accessed data.

Process Management: The swap partition facilitates efficient process management by providing a temporary storage area for swapped-out processes. This enables the system to handle a larger number of running processes and prevents excessive memory usage.

It's worth noting that with the availability of ample RAM in modern systems, the need for a swap partition may vary. Some systems may function adequately without a swap partition or may rely on other alternatives like swap files or dynamic swap allocation. However, in scenarios where memory-intensive applications or heavy multitasking are expected, having a swap partition can be beneficial to maintain system performance and stability.

* 1. Explain LVM

LVM stands for Logical Volume Manager. It is a flexible and advanced disk management system used in Linux and Unix-like operating systems. LVM allows for dynamic allocation, resizing, and management of storage volumes, providing enhanced flexibility and efficiency in handling storage devices.

Here are the key components and concepts of LVM:

Physical Volumes (PV): Physical Volumes are physical storage devices such as hard disk drives (HDDs), solid-state drives (SSDs), or partitions on those devices. LVM combines multiple physical volumes into a single pool of storage.

Volume Groups (VG): Volume Groups are created by grouping one or more physical volumes together. A Volume Group acts as a container or pool that provides the underlying storage for logical volumes.

Logical Volumes (LV): Logical Volumes are created from the available space within a Volume Group. They function as virtual partitions or logical partitions within the Volume Group and can be used as if they were regular disk partitions. Logical Volumes can be resized, moved, or created dynamically.

Extents: Extents are the smallest unit of storage in LVM. They represent fixed-size blocks of storage and can vary in size depending on the configuration. Multiple extents make up a logical volume, and they are allocated from the Volume Group.

Striping and Mirroring: LVM supports striping and mirroring of logical volumes. Striping distributes data across multiple physical volumes, improving read/write performance. Mirroring duplicates data onto multiple physical volumes, providing data redundancy and fault tolerance.

Snapshot: LVM allows the creation of instant read-only snapshots of logical volumes. Snapshots provide a point-in-time copy of a logical volume, which can be useful for backup, testing, or data recovery purposes.

One of the main advantages of LVM is its flexibility in managing storage. It enables administrators to resize logical volumes on the fly, without interrupting the system or requiring manual repartitioning. LVM also provides features such as volume migration, data striping, and mirroring, which enhance performance, reliability, and data availability.

LVM is commonly used in scenarios where there is a need for dynamic storage management, such as servers, virtualization environments, and systems requiring easy resizing and management of disk space.

* 1. Define following terms,
  2. PV

In the context of LVM (Logical Volume Manager), the term "PV" stands for "Physical Volume." Here's a definition of PV within the LVM framework:

Physical Volume (PV):

A Physical Volume refers to a physical storage device or partition that is used as a building block within the LVM infrastructure. It can be a hard disk drive (HDD), solid-state drive (SSD), or a partition on those devices.

When setting up LVM, one or more physical volumes are designated to be included in a Volume Group (VG). The PVs serve as the underlying storage foundation for the logical volumes (LVs) that are created within the VG.

Here are some key points about Physical Volumes (PVs):

PVs can be entire physical storage devices or partitions on those devices. The LVM system recognizes these devices or partitions as individual PVs.

PVs can be added or removed from a Volume Group (VG) dynamically, allowing for easy expansion or contraction of storage resources.

PVs are not directly accessible or usable by the operating system or applications. They need to be included in a Volume Group and further divided into Logical Volumes (LVs) for data storage and management.

PVs can span across multiple disks or partitions, allowing for the creation of larger storage pools by combining physical storage resources.

LVM provides utilities and commands, such as "pvcreate" and "pvdisplay," to initialize and manage Physical Volumes within the LVM setup.

By aggregating multiple Physical Volumes into a Volume Group, LVM provides a layer of abstraction that simplifies disk management and offers flexible allocation and resizing of storage resources. Physical Volumes form an essential component of the LVM architecture, enabling the dynamic allocation and management of logical volumes within the system.

* 1. VG

In the context of LVM (Logical Volume Manager), the term "VG" stands for "Volume Group." Here's a definition of VG within the LVM framework:

Volume Group (VG):

A Volume Group refers to a logical container that consists of one or more Physical Volumes (PVs) in LVM. It serves as a pool of storage space from which Logical Volumes (LVs) can be created.

Here are some key points about Volume Groups (VGs):

VGs are created by combining one or more Physical Volumes (PVs) into a single storage pool. These PVs can be physical storage devices (e.g., hard disk drives, solid-state drives) or partitions on those devices.

The VG represents the underlying storage resource that is used to create and manage Logical Volumes (LVs). LVs are allocated from the available space within the VG.

VGs provide flexibility in allocating and resizing storage resources. PVs can be added or removed from a VG dynamically, allowing for easy expansion or contraction of the storage pool.

VGs have attributes such as a unique name and physical extent size, which determine how the available space is divided among the Logical Volumes.

LVM provides utilities and commands, such as "vgcreate," "vgextend," and "vgdisplay," to create, modify, and display information about Volume Groups within the LVM setup.

Multiple LVs can be created within a Volume Group, each with its own file system and specific size requirements. The allocation and resizing of LVs are managed within the context of the VG.

Volume Groups act as an abstraction layer in LVM, providing a flexible and dynamic storage management framework. By combining multiple PVs into a VG, administrators can efficiently allocate and manage storage resources to meet the needs of applications and systems.

* 1. LV

In the context of LVM (Logical Volume Manager), the term "LV" stands for "Logical Volume." Here's a definition of LV within the LVM framework:

Logical Volume (LV):

A Logical Volume refers to a virtual partition or logical partition created within a Volume Group (VG) in LVM. It represents a logical storage unit that can be used for data storage and management.

Here are some key points about Logical Volumes (LVs):

LVs are created from the available space within a Volume Group (VG). A VG serves as a pool of storage resources from which LVs are allocated.

LVs act as independent block devices and can be treated similar to physical disk partitions. They can be formatted with a file system and mounted to directories to store and access data.

The size of an LV can be easily adjusted (expanded or shrunk) based on the storage requirements, as long as there is sufficient space available within the VG.

Multiple LVs can coexist within a single VG, each with its own unique name and attributes. Each LV can have different sizes, file systems, and configurations.

LVM provides utilities and commands, such as "lvcreate," "lvextend," and "lvdisplay," to create, manage, and display information about Logical Volumes within the LVM setup.

LVs offer features such as mirroring, striping, and snapshots to enhance data protection, performance, and flexibility.

By leveraging Logical Volumes, LVM provides administrators with the ability to manage storage resources in a dynamic and flexible manner. LVs allow for the allocation and resizing of logical partitions based on specific needs, providing a higher level of control and efficiency in storage management compared to traditional fixed-size partitions.

* 1. From which command we can get information about LVM status?

To obtain information about the status of Logical Volume Manager (LVM) in Linux, you can use the "lvs" or "lvdisplay" command. Both commands provide detailed information about the Logical Volumes (LVs) and their status within the LVM setup.

Here's how you can use these commands:

"lvs" Command:

Open a terminal or command prompt and execute the following command:

Copy code

lvs

The "lvs" command without any arguments displays a summary of the LVs in a tabular format. It provides information such as LV name, VG name, LV size, LV status, and LV mount point (if applicable). This command offers a concise overview of the LVs and their current status within the LVM setup.

"lvdisplay" Command:

Open a terminal or command prompt and execute the following command:

Copy code

lvdisplay

The "lvdisplay" command without any arguments provides detailed information about all LVs in the LVM setup. It includes information such as LV name, VG name, LV size, LV UUID, LV status, LV attributes, LV layout, and more. This command provides comprehensive information about the LVs, allowing you to examine their properties and status in detail.

Both "lvs" and "lvdisplay" commands require appropriate permissions (typically superuser or sudo) to access the LVM information. Running these commands as a privileged user will ensure that you have sufficient access to retrieve the LVM status information.

Please note that the specific output and available options may vary slightly depending on the version of LVM and the Linux distribution you are using.

* Task :1
  1. Check current block details

To check the current block details in Linux, you can use the "lsblk" command. The "lsblk" command provides information about block devices, such as disks and partitions, in a tree-like format.

Open a terminal or command prompt and execute the following command:

Copy code

lsblk

The output will display a hierarchical view of block devices and their associated information, including device names, sizes, types, mount points (if applicable), and more. Here's an example of the output:

bash

Copy code

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

sda 8:0 0 500G 0 disk

├─sda1 8:1 0 100G 0 part /

└─sda2 8:2 0 400G 0 part /home

sdb 8:16 0 1TB 0 disk

└─sdb1 8:17 0 1TB 0 part /mnt/data

In this example, "sda" and "sdb" represent two physical disks, and "sda1," "sda2," and "sdb1" represent their respective partitions. The "SIZE" column indicates the size of each block device, and the "MOUNTPOINT" column displays the mount point of any mounted partitions.

By using the "lsblk" command, you can quickly get an overview of the current block devices and their details in your Linux system.

* 1. Mount removable media

To mount removable media, such as USB drives or external storage devices, you can follow these steps:

Insert the removable media into an available USB port on your computer.

Open a terminal or command prompt.

Create a directory (mount point) where you want to mount the removable media. For example, you can create a directory named "usb" under the "/mnt" directory:

bash

Copy code

sudo mkdir /mnt/usb

Use the "lsblk" command to identify the device name of the removable media. The device name typically starts with "/dev/sd" followed by a letter indicating the device identifier (e.g., "/dev/sdb", "/dev/sdc", etc.). Note down the device name of the removable media.

Mount the removable media to the created mount point using the "mount" command. Replace "/dev/sdX" with the actual device name identified in the previous step, and "/mnt/usb" with the path of the mount point you created:

bash

Copy code

sudo mount /dev/sdX /mnt/usb

If the removable media contains a file system that is not automatically detected, you may need to specify the file system type with the "-t" option. For example, to mount a USB drive with a FAT32 file system, you can use the following command:

bash

Copy code

sudo mount -t vfat /dev/sdX /mnt/usb

After executing the mount command, the removable media will be mounted to the specified mount point. You can access its contents by navigating to the mount point directory. For example, to access the contents of the mounted USB drive, you can use:

bash

Copy code

cd /mnt/usb

When you are done using the removable media, you can unmount it to safely remove it from your system. Use the "umount" command followed by the mount point directory:

bash

Copy code

sudo umount /mnt/usb

Following these steps will allow you to mount and access removable media on your Linux system. Remember to adjust the device name and mount point according to your specific setup.

* 1. Unmount removable media

To unmount removable media, such as USB drives or external storage devices, you can follow these steps:

Ensure that you are not actively using any files or applications that are located on the removable media.

Open a terminal or command prompt.

Verify the current mount status of the removable media by using the "mount" command. Look for the entry corresponding to the mount point of the removable media. For example, if the media is mounted at "/mnt/usb", you can run:

perl

Copy code

mount | grep '/mnt/usb'

If the command output shows the removable media as mounted, proceed to unmount it. Use the "umount" command followed by the mount point directory. Replace "/mnt/usb" with the actual mount point of the removable media:

bash

Copy code

sudo umount /mnt/usb

After executing the unmount command, wait for a moment until the process is complete. You will see the command prompt return without any output or error message.

Once the unmount process is complete, it is safe to remove the removable media from your system.

By following these steps, you can safely unmount removable media from your Linux system. Remember to adjust the mount point directory according to the specific location where the media was mounted.

* 1. Create soft link

To create a soft link (symbolic link) in Linux, you can use the "ln" command with the "-s" option. Soft links are used to create shortcuts or references to files or directories.

The basic syntax for creating a soft link is as follows:

php

Copy code

ln -s <target> <link\_name>

Here's how you can create a soft link:

Open a terminal or command prompt.

Navigate to the directory where you want to create the soft link.

Determine the target file or directory for which you want to create the soft link. Ensure that the target file or directory exists.

Use the "ln" command with the "-s" option followed by the target and the desired link name. For example, if you want to create a soft link named "shortcut" to a file named "myfile.txt" located in the same directory, you can run:

bash

Copy code

ln -s myfile.txt shortcut

Once you execute the command, the soft link will be created. You can verify its existence using the "ls" command:

bash

Copy code

ls -l

The output will display the soft link with an arrow indicating its target:

rust

Copy code

shortcut -> myfile.txt

The soft link can now be used to reference or access the target file or directory. Note that if the target file or directory is moved or deleted, the soft link will become broken.

Remember to adjust the target and link name according to your specific use case.

To create a soft link (symbolic link) in Linux, you can use the "ln" command with the "-s" option. Soft links are used to create shortcuts or references to files or directories.

The basic syntax for creating a soft link is as follows:

php

Copy code

ln -s <target> <link\_name>

Here's how you can create a soft link:

Open a terminal or command prompt.

Navigate to the directory where you want to create the soft link.

Determine the target file or directory for which you want to create the soft link. Ensure that the target file or directory exists.

Use the "ln" command with the "-s" option followed by the target and the desired link name. For example, if you want to create a soft link named "shortcut" to a file named "myfile.txt" located in the same directory, you can run:

bash

Copy code

ln -s myfile.txt shortcut

Once you execute the command, the soft link will be created. You can verify its existence using the "ls" command:

bash

Copy code

ls -l

The output will display the soft link with an arrow indicating its target:

rust

Copy code

shortcut -> myfile.txt

The soft link can now be used to reference or access the target file or directory. Note that if the target file or directory is moved or deleted, the soft link will become broken.

Remember to adjust the target and link name according to your specific use case.

* 1. Create hard link

To create a hard link in Linux, you can use the "ln" command without any special options. Hard links create multiple file entries that point to the same underlying data, making them essentially the same file with different names.

The basic syntax for creating a hard link is as follows:

php

Copy code

ln <target> <link\_name>

Here's how you can create a hard link:

Open a terminal or command prompt.

Navigate to the directory where you want to create the hard link.

Determine the target file for which you want to create the hard link. Ensure that the target file exists.

Use the "ln" command followed by the target and the desired link name. For example, if you want to create a hard link named "linkfile" to a file named "myfile.txt" located in the same directory, you can run:

bash

Copy code

ln myfile.txt linkfile

Once you execute the command, the hard link will be created. Both the original file and the hard link point to the same data, so any changes made to one will be reflected in the other.

You can verify the existence of the hard link using the "ls" command:

bash

Copy code

ls -l

The output will display the hard link with the same file size and inode number as the target file.

Hard links do not have separate ownership or permissions from the target file, and they cannot span across different file systems or partitions.

Remember to adjust the target and link name according to your specific use case.

* 1. Show inode number of all files

To display the inode number of all files in a directory, you can use the "ls" command with the "-i" option. The "-i" option shows the inode number along with the file name in the output.

Here's how you can list the inode numbers of files in a directory:

Open a terminal or command prompt.

Navigate to the directory for which you want to display the inode numbers.

Use the "ls" command with the "-i" option followed by the directory path. For example, to display the inode numbers of files in the current directory, you can run:

bash

Copy code

ls -i

The command will list the files in the directory along with their respective inode numbers. Each line will display the inode number followed by the file name.

Here's an example output:

Copy code

134206 myfile.txt

134207 myimage.jpg

134208 myscript.sh

In this example, "myfile.txt" has an inode number of 134206, "myimage.jpg" has 134207, and "myscript.sh" has 134208.

By using the "ls" command with the "-i" option, you can easily view the inode numbers of files in a directory.

* 1. Find a file is equal to 10 mb

To find a file that is exactly 10 MB in size, you can use the "find" command along with the appropriate size specifier. The size specifier uses the unit "c" for bytes, "k" for kilobytes, "M" for megabytes, and "G" for gigabytes.

Here's how you can find a file with a size of 10 MB:

Open a terminal or command prompt.

Navigate to the directory where you want to start the search or specify the root directory for a system-wide search.

Use the "find" command with the "-size" option followed by the size specifier and the size you are looking for. For a file with exactly 10 MB, you can use:

arduino

Copy code

find . -size 10M

This command will search for files starting from the current directory (denoted by "."), with a size of 10 MB.

After executing the command, the "find" command will list any files that match the specified size criterion. If there is a file exactly 10 MB in size, its path and other information will be displayed.

Please note that the search operation may take some time depending on the size of the directory or the scope of the search.

Adjust the directory path and size specifier according to your specific requirements. You can replace "." with the desired directory path or use "/" for a system-wide search. Additionally, you can modify the size specifier to search for files of different sizes (e.g., 10k for 10 kilobytes, 10G for 10 gigabytes, etc.).

* 1. Find a file which have more then 10 mb

To find files that are larger than 10 MB in size, you can use the "find" command with the appropriate size specifier. The size specifier uses the unit "c" for bytes, "k" for kilobytes, "M" for megabytes, and "G" for gigabytes.

Here's how you can find files larger than 10 MB:

Open a terminal or command prompt.

Navigate to the directory where you want to start the search or specify the root directory for a system-wide search.

Use the "find" command with the "-size" option and a size specifier indicating greater than 10 MB. For example, to find files larger than 10 MB, you can use:

arduino

Copy code

find . -size +10M

This command will search for files starting from the current directory (denoted by "."), with a size larger than 10 MB.

After executing the command, the "find" command will list any files that match the specified size criterion. The output will include the paths and other information of the files that are larger than 10 MB.

Please note that the search operation may take some time depending on the size of the directory or the scope of the search.

Adjust the directory path and size specifier according to your specific requirements. You can replace "." with the desired directory path or use "/" for a system-wide search. Additionally, you can modify the size specifier to search for files of different sizes (e.g., +5k for files larger than 5 kilobytes, +2G for files larger than 2 gigabytes, etc.).

* 1. Find directory list

To find a list of directories in a specific location, you can use the "ls" command with the "-d" option and the "\*" wildcard character. This will display only the directories and exclude files from the list.

Here's how you can find the directory list:

Open a terminal or command prompt.

Navigate to the location where you want to find the directory list.

Use the "ls" command with the "-d" option and the "\*" wildcard character. For example:

bash

Copy code

ls -d \*/

This command will list only the directories in the current location.

After executing the command, the "ls" command will display the names of the directories in the specified location.

Here's an example output:

Copy code

directory1/ directory2/ directory3/

In this example, "directory1", "directory2", and "directory3" are the directories listed.

By using the "ls" command with the "-d" option and the "\*" wildcard character, you can easily find the directory list in a specific location.

* 1. Find file list

To find a list of directories in a specific location, you can use the "ls" command with the "-d" option and the "\*" wildcard character. This will display only the directories and exclude files from the list.

Here's how you can find the directory list:

Open a terminal or command prompt.

Navigate to the location where you want to find the directory list.

Use the "ls" command with the "-d" option and the "\*" wildcard character. For example:

bash

Copy code

ls -d \*/

This command will list only the directories in the current location.

After executing the command, the "ls" command will display the names of the directories in the specified location.

Here's an example output:

Copy code

directory1/ directory2/ directory3/

In this example, "directory1", "directory2", and "directory3" are the directories listed.

By using the "ls" command with the "-d" option and the "\*" wildcard character, you can easily find the directory list in a specific location.

* 1. Find soft link list

To find a list of soft links (symbolic links) in a specific location, you can use the "find" command with the "-type l" option. This will search for files of type symbolic link and provide you with a list of soft links.

Here's how you can find the soft link list:

Open a terminal or command prompt.

Navigate to the location where you want to find the soft link list.

Use the "find" command with the "-type l" option. For example:

lua

Copy code

find . -type l

This command will search for soft links in the current location (denoted by ".") and its subdirectories.

After executing the command, the "find" command will display the names and paths of the soft links in the specified location.

Here's an example output:

bash

Copy code

./link1

./link2

./subdirectory/link3

In this example, "link1", "link2", and "link3" are the soft links listed. The paths may include subdirectories if the soft links are located within them.

By using the "find" command with the "-type l" option, you can easily find the list of soft links in a specific location.

* Task:2
  1. Create a new partition with following requirements, Size 1G File type xfs

To create a new partition with a size of 1GB and the XFS file system type, you can use the following steps:

Note: Creating a new partition involves modifying disk partitions, which can potentially result in data loss. It is essential to take appropriate precautions and have a backup of your important data before proceeding.

Open a terminal or command prompt.

Identify the disk on which you want to create the new partition. You can use the "lsblk" or "fdisk -l" command to list the available disks and their current partitions.

Use a partitioning tool such as "fdisk" or "parted" to create the new partition. For this example, we'll use "parted".

Run the following command to start the partitioning tool:

bash

Copy code

sudo parted /dev/sdX

Replace "/dev/sdX" with the appropriate disk identifier for your system.

Once you are in the partitioning tool, follow these steps:

a. Create a new partition table if one does not exist already. This step is necessary if the disk does not have any partitions yet. You can use the "mklabel" command with the desired partition table type. For example, to create an MBR partition table, you can run:

scss

Copy code

(parted) mklabel msdos

If the disk already has a partition table, skip this step.

b. Create a new partition by using the "mkpart" command followed by the file system type, starting point, and end point. In this case, we want to create a 1GB partition using the XFS file system type. Run the following command:

scss

Copy code

(parted) mkpart primary xfs 0% 1000MB

This command creates a primary partition starting from 0% (beginning of the disk) and ending at 1000MB.

c. Set the partition flag if needed. Depending on your requirements, you may need to set a specific partition flag using the "set" command. For example, to set the boot flag for the partition, you can run:

csharp

Copy code

(parted) set 1 boot on

Adjust the partition number accordingly if you created multiple partitions.

d. Exit the partitioning tool by typing "quit". This will save the changes and exit the tool.

After creating the partition, you need to format it with the XFS file system.

Run the following command to format the new partition:

bash

Copy code

sudo mkfs.xfs /dev/sdX1

Replace "/dev/sdX1" with the appropriate partition identifier for your system. Ensure you specify the correct partition number if you created multiple partitions.

The new partition with the XFS file system is now created and formatted. You can mount it to a directory of your choice using the "mount" command and add an entry to the "/etc/fstab" file for automatic mounting during system startup.

For example, to mount the new partition to the "/mnt/newpartition" directory, run the following command:

bash

Copy code

sudo mount /dev/sdX1 /mnt/newpartition

To make the mount persistent, add an entry to the "/etc/fstab" file. Open the file with a text editor, such as:

bash

Copy code

sudo nano /etc/fstab

Add the following entry at the end of the file:

bash

Copy code

/dev/sdX1 /mnt/newpartition xfs defaults 0 0

Replace "/dev/sdX1" with the appropriate partition identifier and "/mnt

* 1. Create three primary partitions

To create three primary partitions on a disk, you can use a partitioning tool like "fdisk" or "parted". Here's a general outline of the steps using the "fdisk" command:

Note: Creating or modifying partitions involves disk operations that can result in data loss. Ensure you have a backup of your important data before proceeding.

Open a terminal or command prompt.

Identify the disk on which you want to create the partitions. You can use the "lsblk" or "fdisk -l" command to list the available disks and their current partitions.

Run the following command to start the partitioning tool for the specified disk:

bash

Copy code

sudo fdisk /dev/sdX

Replace "/dev/sdX" with the appropriate disk identifier for your system.

Once you are in the partitioning tool, follow these steps to create three primary partitions:

a. Check the existing partitions, if any, by using the "p" command.

b. Create the first partition by using the "n" command:

Select "p" for primary partition type.

Choose the starting and ending sector for the partition. If you want to use the entire disk for the first partition, you can press Enter to accept the default values.

c. Create the second partition by using the "n" command:

Select "p" for primary partition type.

Choose the starting and ending sector for the partition. Ensure the starting sector follows the end of the first partition.

d. Create the third partition following the same steps as above.

e. Once you have created all three partitions, you can review them by using the "p" command to ensure they are correctly set up.

f. Save the changes and exit the partitioning tool by using the "w" command.

After creating the partitions, you can format them with the desired file systems using the appropriate formatting commands such as "mkfs.ext4" for ext4 file system or "mkfs.xfs" for XFS file system.

For example, to format the first partition as ext4, you can run:

bash

Copy code

sudo mkfs.ext4 /dev/sdX1

Replace "/dev/sdX1" with the appropriate partition identifier.

Repeat the formatting step for the other partitions, using the respective partition identifiers.

The three primary partitions are now created and formatted. You can mount them to directories of your choice using the "mount" command and add entries to the "/etc/fstab" file for automatic mounting during system startup.

Remember to adjust the partition identifiers ("/dev/sdX1", "/dev/sdX2", etc.) and file system types according to your specific requirements.

* 1. Mount new partitions via UUID and LABLE

To mount new partitions using UUID or LABEL instead of device identifiers, you can follow these steps:

Find the UUID or LABEL of the partitions you want to mount. Run the following command to list the available partitions and their UUIDs and labels:

css

Copy code

lsblk -o NAME,UUID,LABEL

Note down the UUID or LABEL of the partitions you want to mount.

Create mount points (directories) where you want to mount the partitions. For example, you can create directories named "/mnt/partition1", "/mnt/partition2", etc. Use the following command to create the directories:

bash

Copy code

sudo mkdir /mnt/partition1 /mnt/partition2

Adjust the directory names according to your preferences.

Open the "/etc/fstab" file with a text editor using the following command:

bash

Copy code

sudo nano /etc/fstab

Add entries to the "/etc/fstab" file for each partition you want to mount. Use either the UUID or LABEL options. Here are two examples:

Using UUID:

javascript

Copy code

UUID=<UUID> /mnt/partition1 ext4 defaults 0 0

Replace "<UUID>" with the actual UUID of the partition.

Using LABEL:

css

Copy code

LABEL=<LABEL> /mnt/partition2 ext4 defaults 0 0

Replace "<LABEL>" with the actual label of the partition.

Adjust the file system type (e.g., "ext4") and mount options (e.g., "defaults") as per your specific requirements.

Save the changes and exit the text editor.

css

Copy code

sudo mount -a

This will mount the partitions specified in the "/etc/fstab" file using their UUID or LABEL.

Now, the new partitions should be mounted to the specified directories using UUID or LABEL instead of device identifiers.

* 1. Create “swap” partition

To create a "swap" partition, which is used for virtual memory in Linux systems, you can follow these steps:

Note: Creating or modifying disk partitions can result in data loss. Ensure you have a backup of your important data before proceeding.

Open a terminal or command prompt.

Identify the disk on which you want to create the swap partition. You can use the "lsblk" or "fdisk -l" command to list the available disks and their current partitions.

Once you have identified the disk, use a partitioning tool such as "fdisk" or "parted" to create the swap partition. For this example, we'll use "fdisk".

Run the following command to start the partitioning tool for the selected disk:

bash

Copy code

sudo fdisk /dev/sdX

Replace "/dev/sdX" with the appropriate disk identifier for your system.

Once you are in the partitioning tool, follow these steps:

a. Create a new partition by using the "n" command. You can choose either a primary or logical partition based on your disk's partitioning scheme.

b. Specify the partition size. For a swap partition, you typically allocate it the desired size. For example, to create a 2GB swap partition, you can specify the size as "+2G".

c. Set the partition type to "Linux swap". Use the "t" command to change the partition type, and enter the code for Linux swap, which is "82".

d. Write the changes to the disk by using the "w" command. This will save the changes and exit the tool.

After creating the swap partition, you need to initialize it as a swap space.

Run the following command to initialize the swap partition:

bash

Copy code

sudo mkswap /dev/sdX1

Replace "/dev/sdX1" with the appropriate partition identifier for your system.

Activate the swap partition by using the "swapon" command:

bash

Copy code

sudo swapon /dev/sdX1

Replace "/dev/sdX1" with the appropriate partition identifier for your system.

To make the swap partition automatically activated at system startup, you need to add an entry to the "/etc/fstab" file.

Open the "/etc/fstab" file with a text editor using the following command:

bash

Copy code

sudo nano /etc/fstab

Add the following entry at the end of the file:

bash

Copy code

/dev/sdX1 none swap sw 0 0

Replace "/dev/sdX1" with the appropriate partition identifier for your system.

Save the changes and exit the text editor.

Now, the swap partition is created, initialized, and activated. It will be automatically mounted as swap during system startup based on the entry in the "/etc/fstab" file.

* Task: 3
  1. Create new LVM

To create a new Logical Volume Management (LVM) setup, including Physical Volumes (PV), Volume Group (VG), and Logical Volumes (LV), you can follow these steps:

Note: Modifying disk partitions and setting up LVM can potentially result in data loss. Ensure you have a backup of your important data before proceeding.

Open a terminal or command prompt.

Identify the disks or partitions you want to use as Physical Volumes (PVs) for LVM. You can use the "lsblk" or "fdisk -l" command to list the available disks and their current partitions.

Create Physical Volumes (PVs) on the selected disks or partitions. Use the following command for each disk or partition:

bash

Copy code

sudo pvcreate /dev/sdX

Replace "/dev/sdX" with the appropriate disk or partition identifier for your system.

Once you have created the Physical Volumes (PVs), you can create a Volume Group (VG) that encompasses them.

Run the following command to create a Volume Group:

bash

Copy code

sudo vgcreate myvg /dev/sdX1 /dev/sdY1

Replace "myvg" with the desired name for your Volume Group, and "/dev/sdX1 /dev/sdY1" with the appropriate Physical Volume names you created.

After creating the Volume Group (VG), you can create Logical Volumes (LVs) within it.

Run the following command to create a Logical Volume:

Copy code

sudo lvcreate -n mylv -L 10G myvg

Replace "mylv" with the desired name for your Logical Volume, "10G" with the desired size for the volume, and "myvg" with the name of your Volume Group.

You can repeat the previous step to create additional Logical Volumes within the same Volume Group if needed.

After creating Logical Volumes, you need to format them with a file system of your choice.

Run the following command to format a Logical Volume with a file system:

bash

Copy code

sudo mkfs.ext4 /dev/myvg/mylv

Replace "/dev/myvg/mylv" with the appropriate path to your Logical Volume.

Once the Logical Volume is formatted, you can mount it to a directory of your choice using the "mount" command.

Run the following command to mount the Logical Volume:

bash

Copy code

sudo mount /dev/myvg/mylv /mnt/mylv

Replace "/dev/myvg/mylv" with the appropriate path to your Logical Volume, and "/mnt/mylv" with the desired mount point.

Now, you have created a new LVM setup with Physical Volumes, a Volume Group, and Logical Volumes. The Logical Volumes are formatted and mounted, ready for use.

* 1. Extend this new LVM

To extend an existing Logical Volume Management (LVM) setup by adding more space to a Volume Group (VG) and extending a Logical Volume (LV), you can follow these steps:

Note: Modifying disk partitions and extending LVM can potentially result in data loss. Ensure you have a backup of your important data before proceeding.

Identify the disk or partition that you want to add as a Physical Volume (PV) to extend the Volume Group. You can use the "lsblk" or "fdisk -l" command to list the available disks and their current partitions.

Create a Physical Volume (PV) on the selected disk or partition. Use the following command:

bash

Copy code

sudo pvcreate /dev/sdX

Replace "/dev/sdX" with the appropriate disk or partition identifier for the space you want to add.

Once you have created the Physical Volume (PV), you can extend the Volume Group (VG) to include it.

Run the following command to extend the Volume Group:

bash

Copy code

sudo vgextend myvg /dev/sdX1

Replace "myvg" with the name of your existing Volume Group, and "/dev/sdX1" with the appropriate Physical Volume name you created.

After extending the Volume Group, you can extend the Logical Volume (LV) to use the additional space.

Run the following command to extend the Logical Volume:

bash

Copy code

sudo lvextend -r -L +10G /dev/myvg/mylv

Replace "/dev/myvg/mylv" with the appropriate path to your Logical Volume, and "+10G" with the amount of space you want to add. The "-r" option automatically resizes the file system within the Logical Volume.

If you don't want to resize the file system immediately, omit the "-r" option.

Verify the new size of the Logical Volume using the following command:

bash

Copy code

sudo lvdisplay /dev/myvg/mylv

Ensure that the "LV Size" reflects the desired size.

If you didn't resize the file system in the previous step, you need to manually resize it.

Run the appropriate command based on the file system type. For example, if the file system is ext4, use the following command:

bash

Copy code

sudo resize2fs /dev/myvg/mylv

Replace "/dev/myvg/mylv" with the appropriate path to your Logical Volume.

Now, your Logical Volume has been extended and the file system (if resized) is ready to use the additional space.

Remember to adjust the commands and paths based on your specific LVM setup, Volume Group name, Logical Volume name, and disk or partition identifiers.