Linux- File System Management Lab :

#### **Objective 1** : Learn how to partition a disk, create filesystems, and mount them.

Outcome:

This lab will help you gain hands-on experience in disk partitioning, file system creation, and mounting partitions on Linux.

1. Partition a Disk:

# Use fdisk or parted to create partitions on a disk (e.g., /dev/sdb).

# Create a primary partition and a swap partition.

# Use lsblk and fdisk -l to confirm the new partitions.

2. Create File Systems:

# Format the partitions with different file systems (e.g., ext4, xfs, btrfs) using the mkfs command.

# Check the file system using fsck.

3. Mount Partitions:

# Mount the new partitions manually using mount (e.g., mount /dev/sdb1 to /mnt/data).

# Add entries to /etc/fstab to ensure automatic mounting on boot.

4. Verify and Access:

# Use df -h to check mounted file systems and disk usage.

# Access files from the new mount point and test read/write operations.

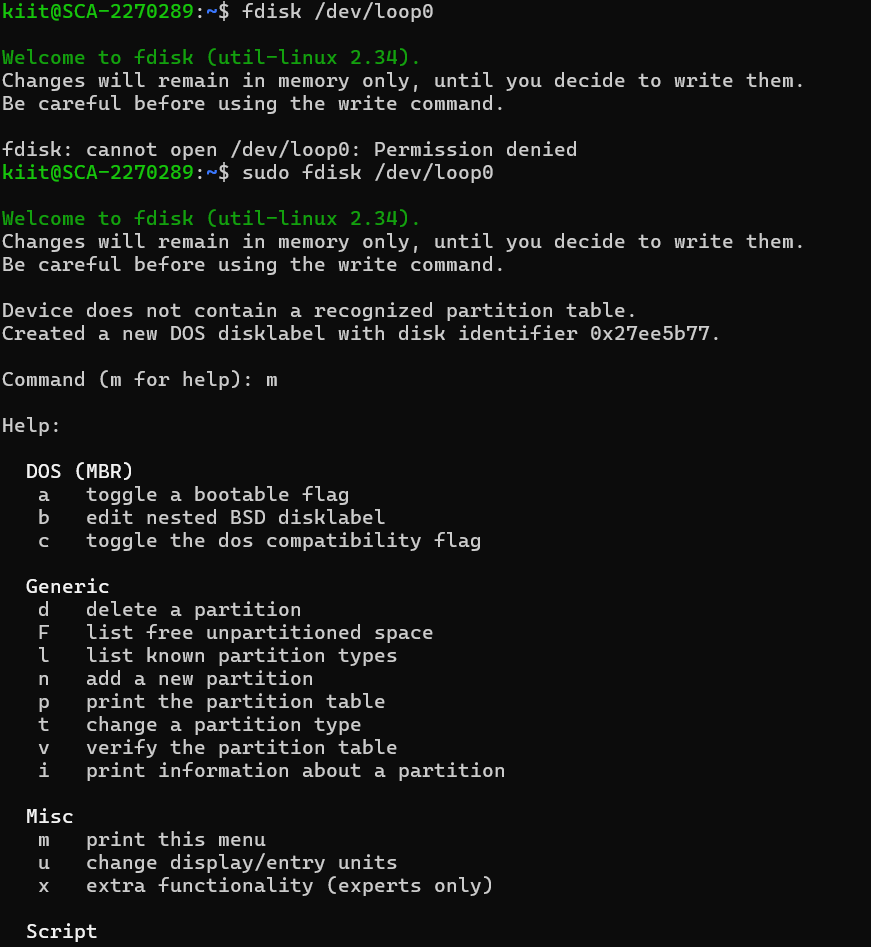
**Solution :**

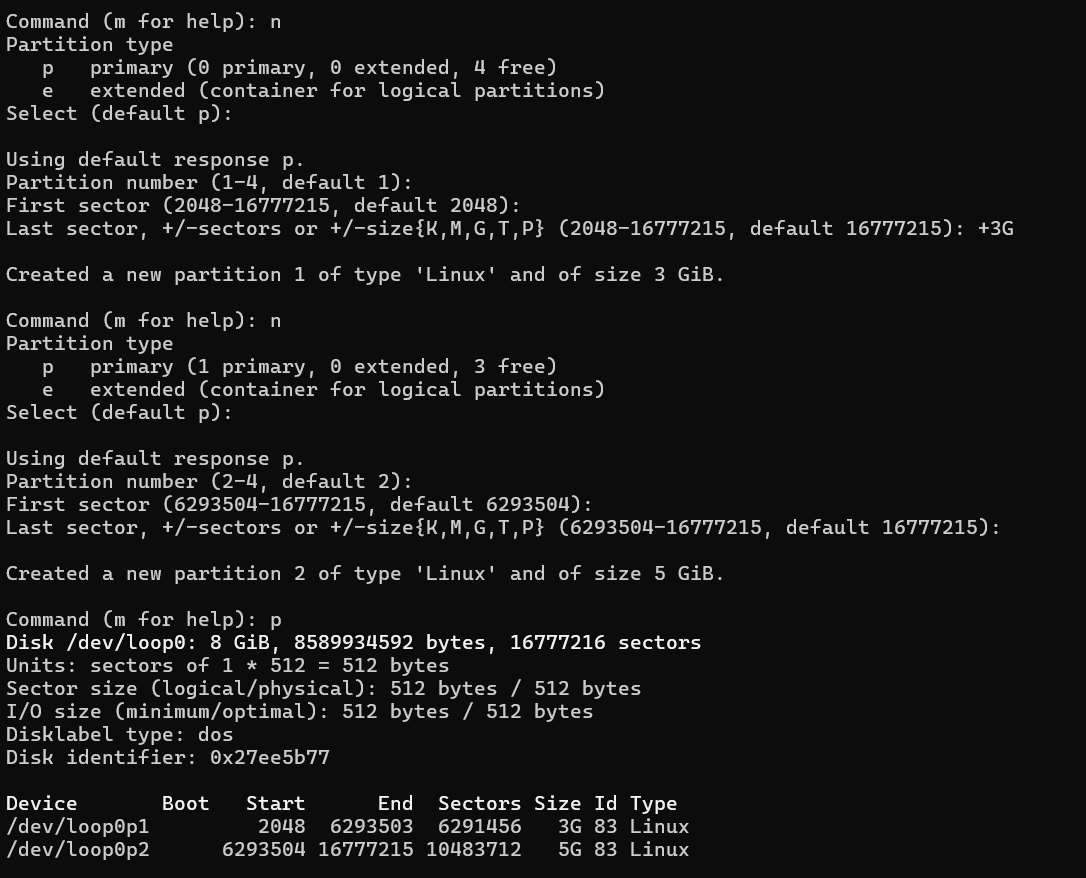
To create extra space I have used command :

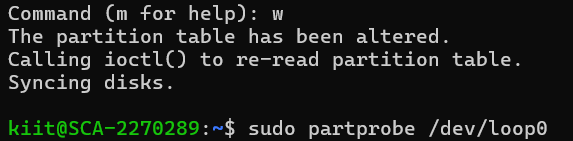
**dd if=/dev/zero of=~/2gb\_disk.img bs=1M count=2048**

**sudo losetup -fP ~/2gb\_disk.img**

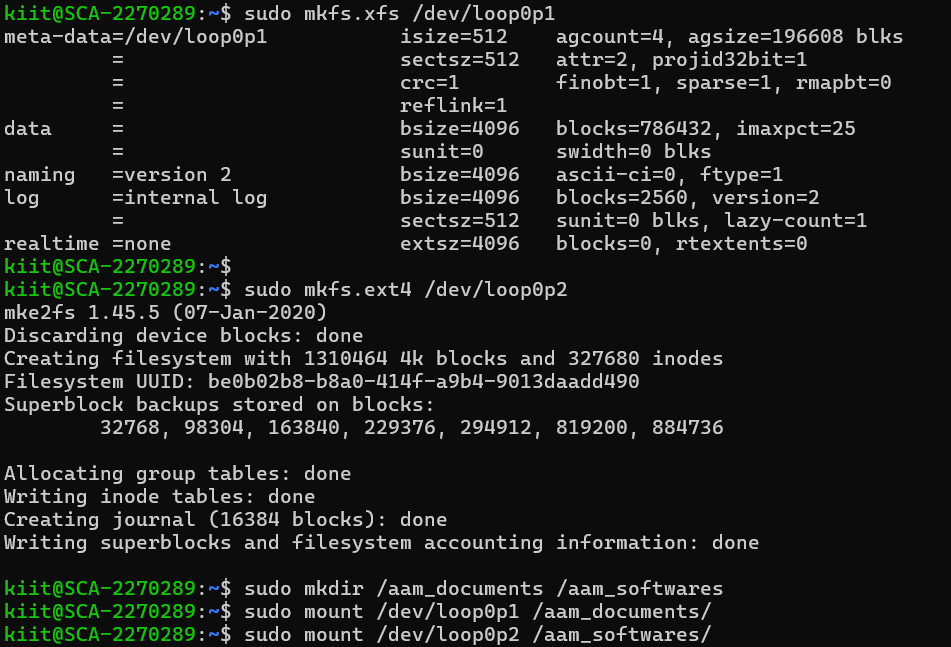
# above one will create space of 2 gb but I have created 8b gb space

**Step : **

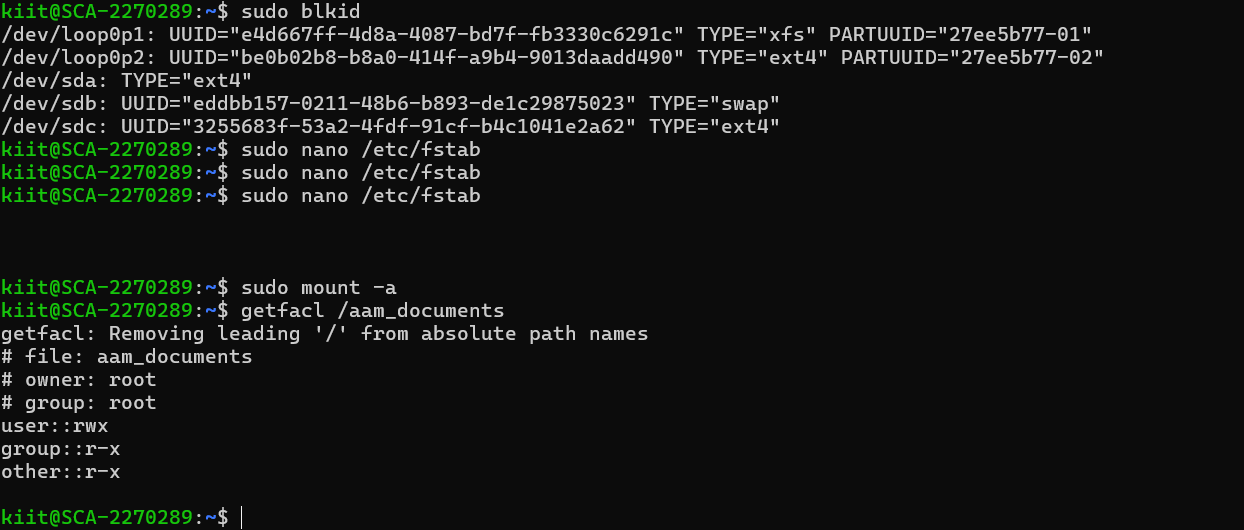
****

****

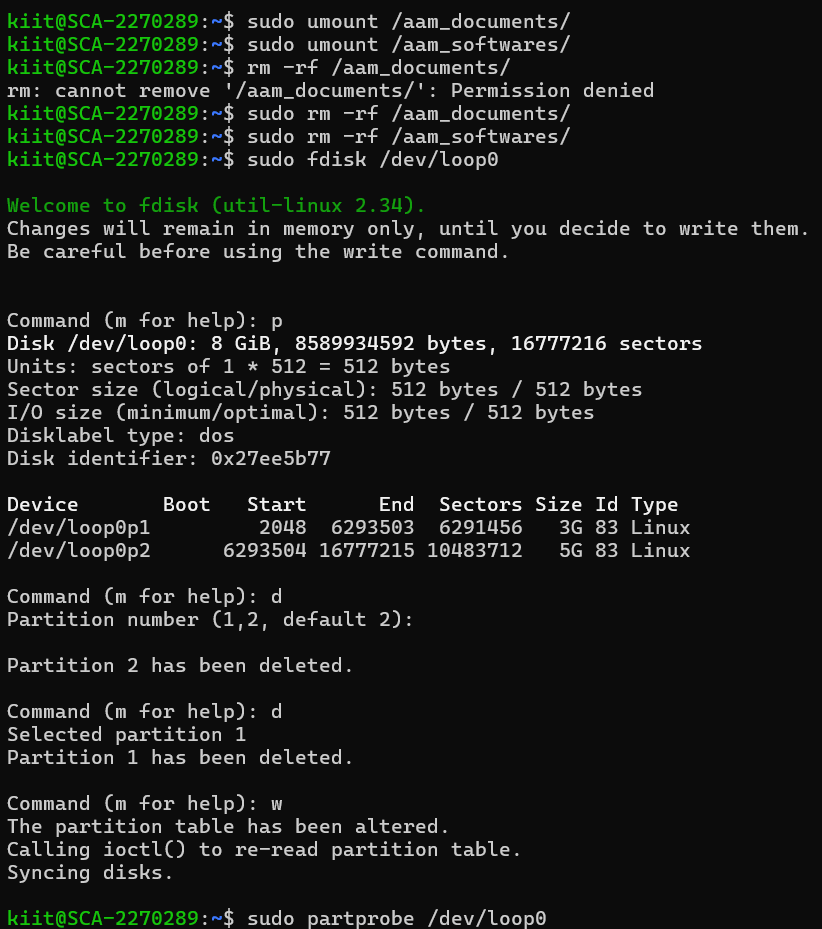
****

****

** #** to see wheather it is mounted or not.

****

## To unmount and delete 2 folder i.e aam\_document and aam\_softwares and deleting 2 created partition :



# **Objective 2**: Practice managing directories and controlling permissions in a Linux file system.

**Outcome** : This lab will help you manage directories, control access to them, and work with advanced permissions and ACLs.

1. Create Directories:

# Use the mkdir command to create a complex directory structure (e.g., /home/user/docs, /home/user/projects).

2. Set Permissions:

# Use chmod to set permissions for different directories and files. For example, set read/write/execute permissions for the owner, group, and others.

# Use chown to change ownership of files and directories.

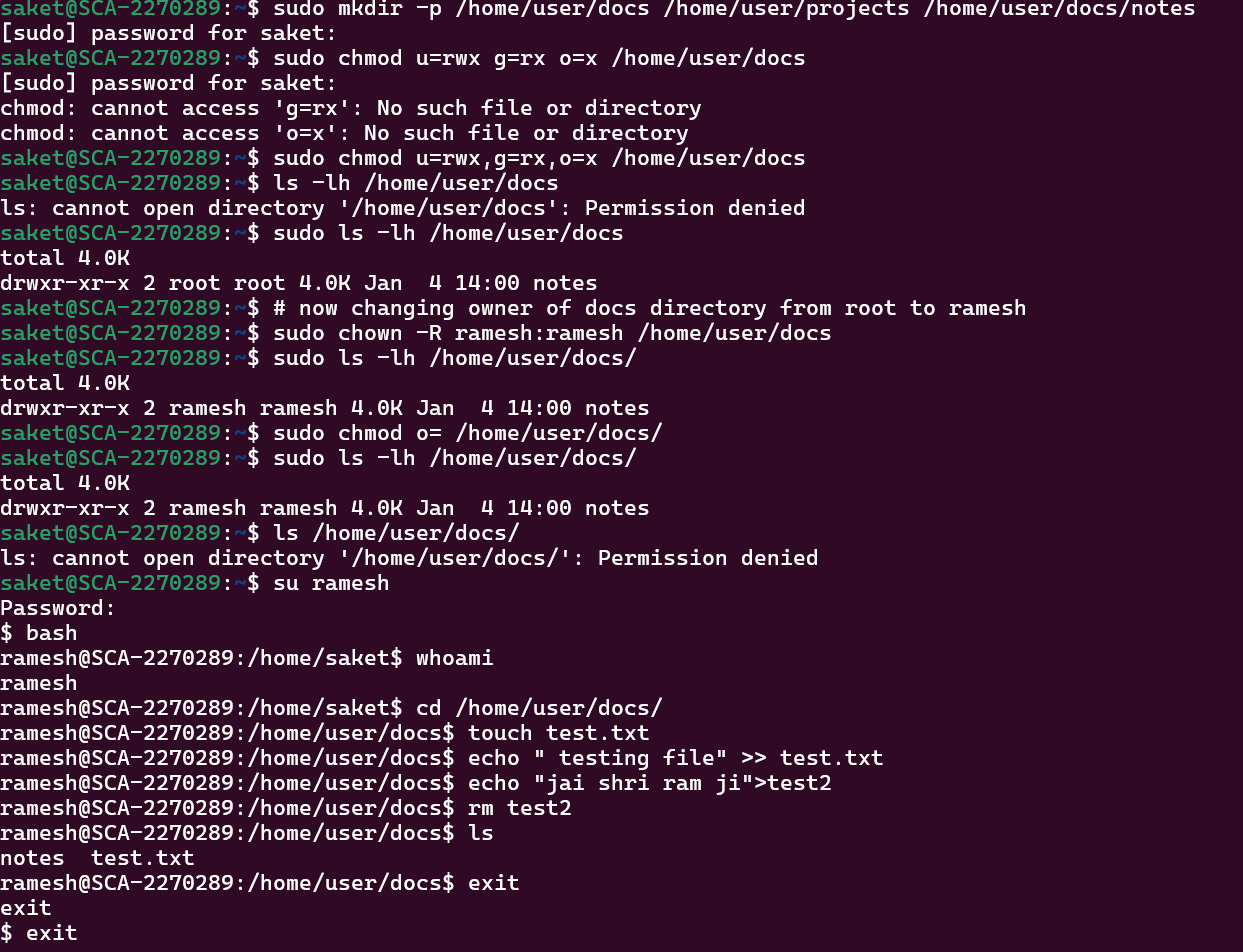
3. Test Directory Permissions:

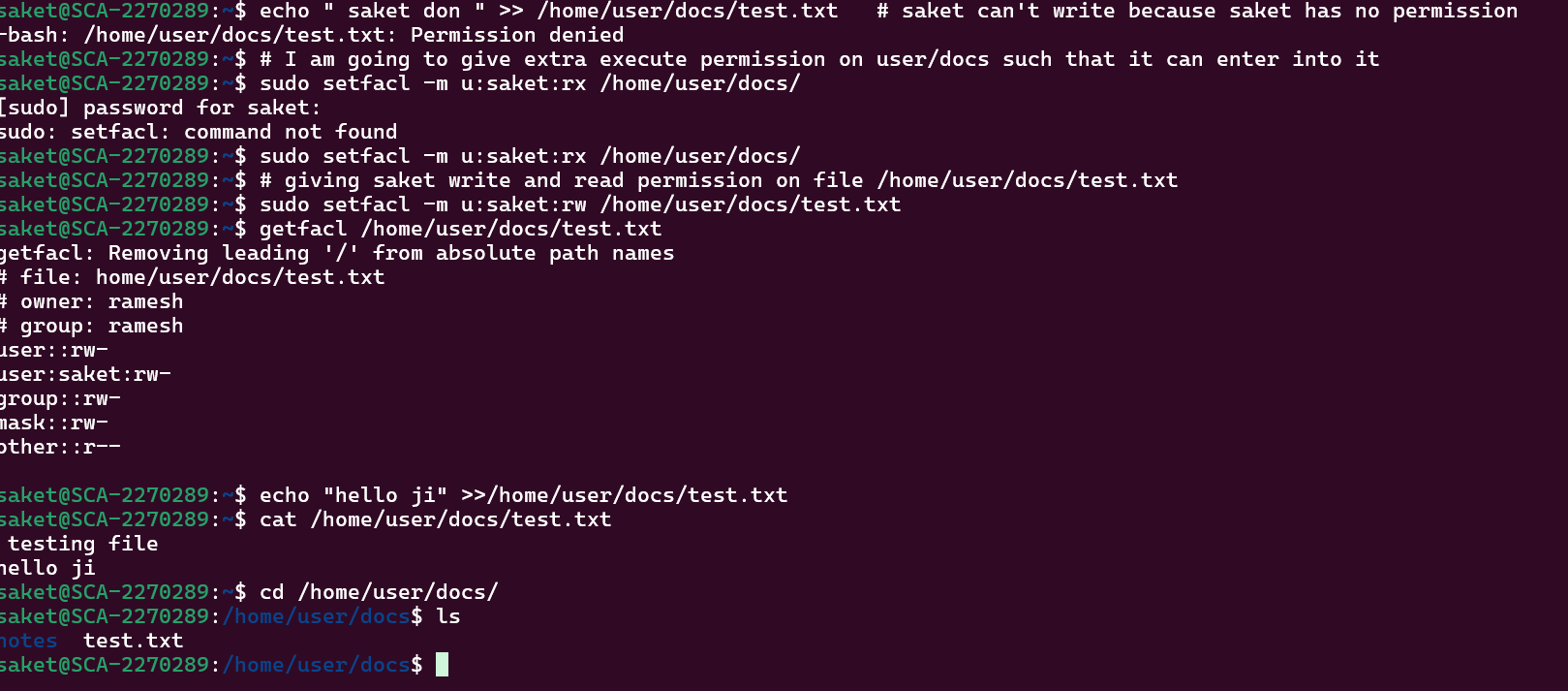
# Ensure that users without proper permissions cannot access directories.

# Test creating, deleting, and modifying files inside these directories.

4. Use Access Control Lists (ACLs):

# Use setfacl to set additional ACLs for files and directories, allowing more fine-grained control over file access.





# Objective 3: Set up and mount a Network File System (NFS) to share files between two Linux machines.

Outcome : This lab will teach you how to configure and use NFS for sharing files across multiple systems, a critical task for centralized storage in Linux environments.

1. Install NFS Server:

# Install and configure the NFS server on a Linux machine using apt-get or yum.

# Edit /etc/exports to specify which directories are shared (e.g., /mnt/data).

2. Configure NFS Server:

# Export the shared directory using the exportfs command.

# Start the NFS service with systemctl start nfs-server.

3. Mount NFS on Client:

# On another Linux machine, mount the shared directory using the mount command (e.g., mount <server\_ip>:/mnt/data /mnt/nfs).

4. Verify NFS Functionality:

# Test file creation and modification across the network to verify that NFS is functioning correctly.

**Solution :**

# NFS (Network file system) :

Using this, a user on the client computer can easily access the files/folders on server side like as they are accessing locally.

🡪 In short, two different machine connected on a same network then one machine i.e client can access the file/folder present on other machine i.e server with help of NFS. If we do any changes in the shared file/directory from client side then changes is also visible or saved in server side and vice-versa. Server give permission on file/folder for accessing to client by ip address or host name of client, we will see it

## NFS Sever side configuration :

To install NFS Packages :

**yum install nfs-utils libnfsidmap**

To enable and start the nfs services.

**Systemctl enable rpcbind,nfs-server**

**systemctl start rpcbind.service,nfs-server.service,**

**rpc-statd.service,nfs-idmap.service**

**🡪 T**o check status wheather it is started or not write status in place of start only in above line.

**Step** : Create a directory (i.e shared among clients) and give all the permissions to this directory i.e rwx for user,group,other

e.g : mkdir /server/apps

**Step :** Modify the /etc/exports file on server side and insert a command given below. To connect with client on this directory or any directoy for sharing with clients

Syntax : **/apps <client\_ip\_address>(rw,sync, no\_root\_squash)**

**#** Means /apps folder will be shared only with clint with specified ip\_address, rw permission is given on /apps directory, sync means changes made in /apps will be visible in both side, no\_root\_squash means no root or sudo permission required .

**/apps \* (rw,sync, no\_root\_squash) #** \* Means all client present on network can acees /apps.

**Step : exportfs -rv #** For saving the changes in /etc/exports

## NFS Client Side configuration :

To install NFS Packages :

**yum install nfs-utils rpcbind**

Enable and start rpcbind service :

**Systemctl start rpcbind**

Step : Show mount or to connect with NFS Server use command :

**showmount -e <ip address of server side>**

**Step :** Create A directory to link with shared directory at server side.

Or create a mount point(i.e a directory) : **e.g mkdir /mnt/apps**

**Step:**  Mount the NFS File system

**mount <ip\_address\_server> : /server/apps /mnt/apps**

**🡪** Connection established successfully now create a file in /server/apps that file will be also created or visible in /mnt/apps in client side and vice versa is also true.

# **Objective 4: Learn to analyze disk usage and clean up disk space by removing unnecessary files.**

1. Check Disk Usage:

# Use the df -h command to check the disk space usage of the file system.

# Use du -sh <directory> to check the size of specific directories.

2. Find Large Files:

# Use find / -type f -size +100M to locate files larger than 100MB.

# Use ncdu to interactively view and navigate through disk usage.

3. Clean Up Old Files:

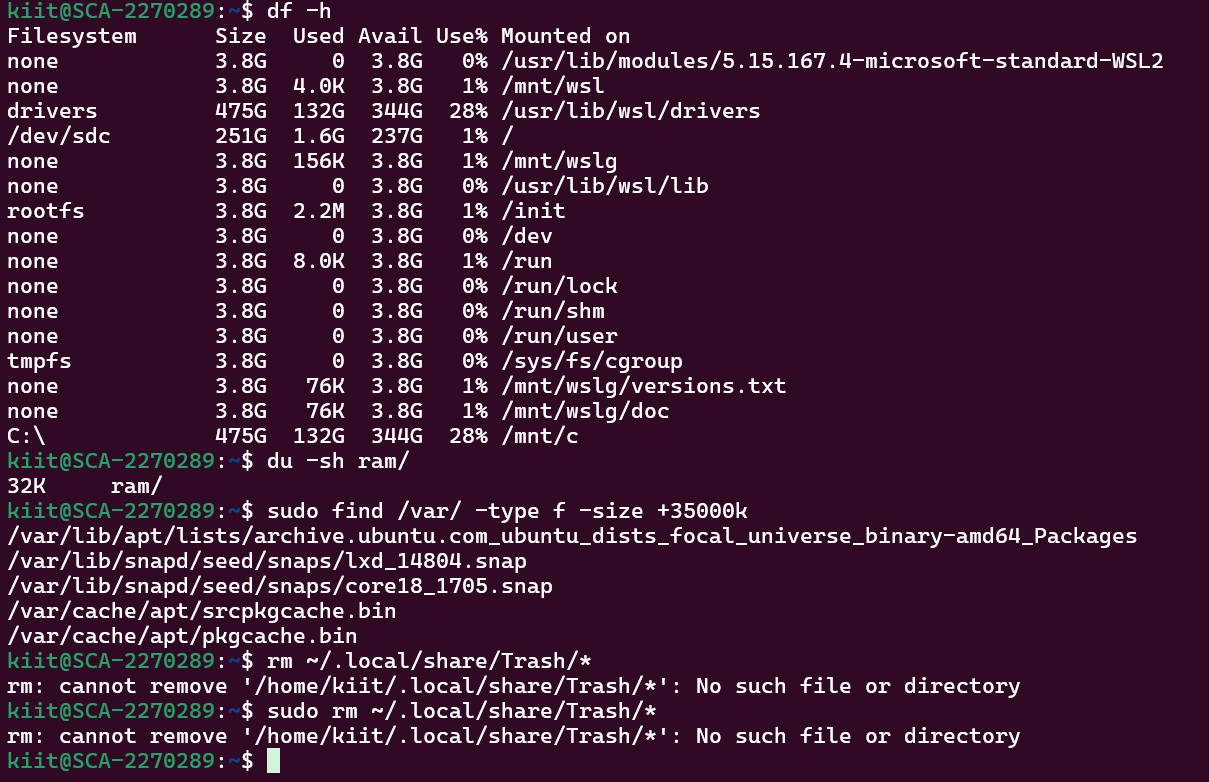
# Identify and delete unnecessary files using the rm command.

# Empty the trash using rm -rf ~/.local/share/Trash/\*.

4. Automate Cleanup:

# Set up a cron job to automate cleanup tasks like deleting old log files or temporary files.

**Solution :**

****

# Objective 5: Set up and manage logical volumes for flexible disk space management.

**Outcome :**  You will learn how to create, manage, and resize logical volumes using LVM, which is a flexible method for managing disk space in Linux.

1. Create Physical Volume (PV):

# Use pvcreate to initialize a physical volume on a disk (e.g., /dev/sdb).

2. Create Volume Group (VG):

# Use vgcreate to create a volume group (e.g., vg\_data).

3. Create Logical Volume (LV):

# Use lvcreate to create a logical volume from the volume group (e.g., lv\_data).

4. Create File System:

# Format the logical volume with a file system (e.g., mkfs.ext4 /dev/vg\_data/lv\_data).

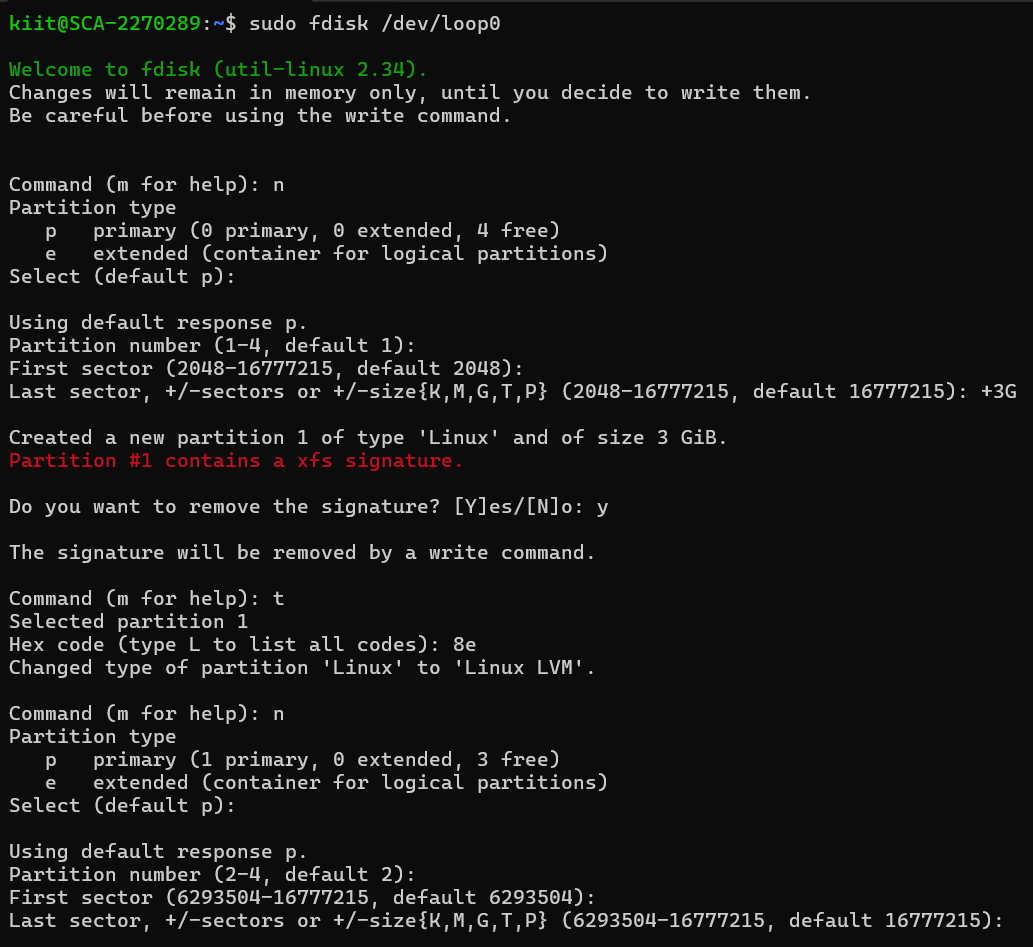
5. Mount and Extend Logical Volume:

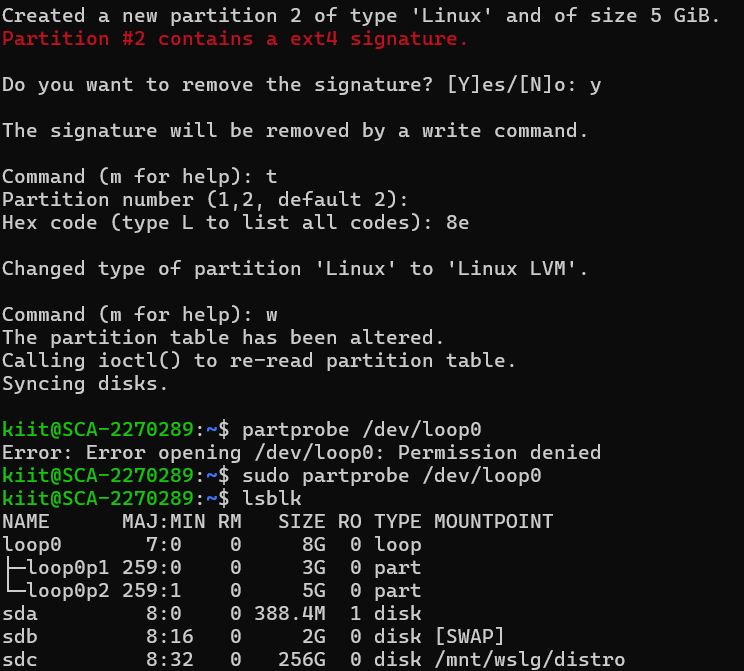
# Mount the logical volume and use lvextend to increase its size as needed.

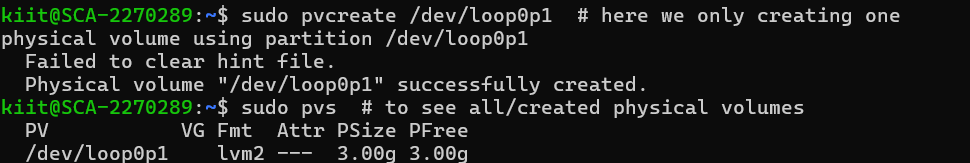
6. Resize File System:

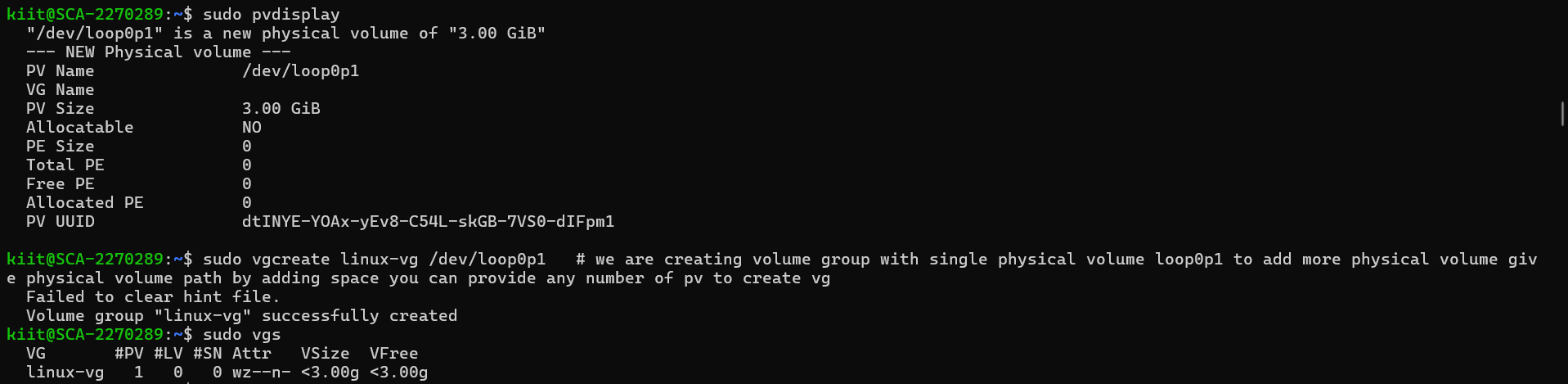
# Use resize2fs or xfs\_growfs to resize the file system after extending the logical volume.

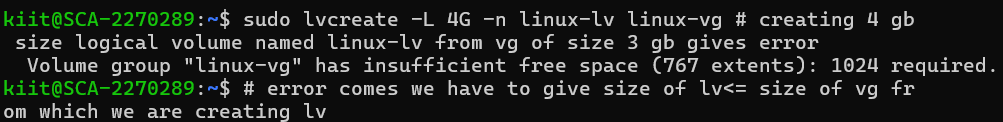
**Solution :**

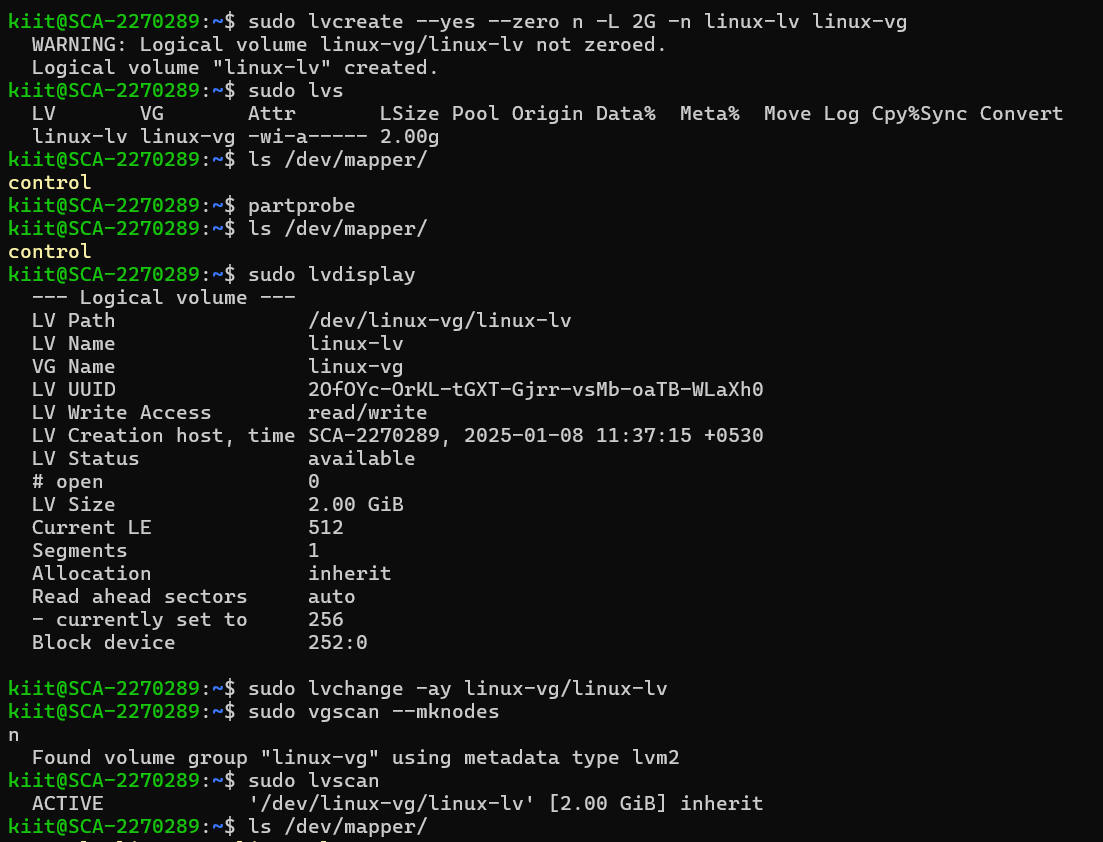
****

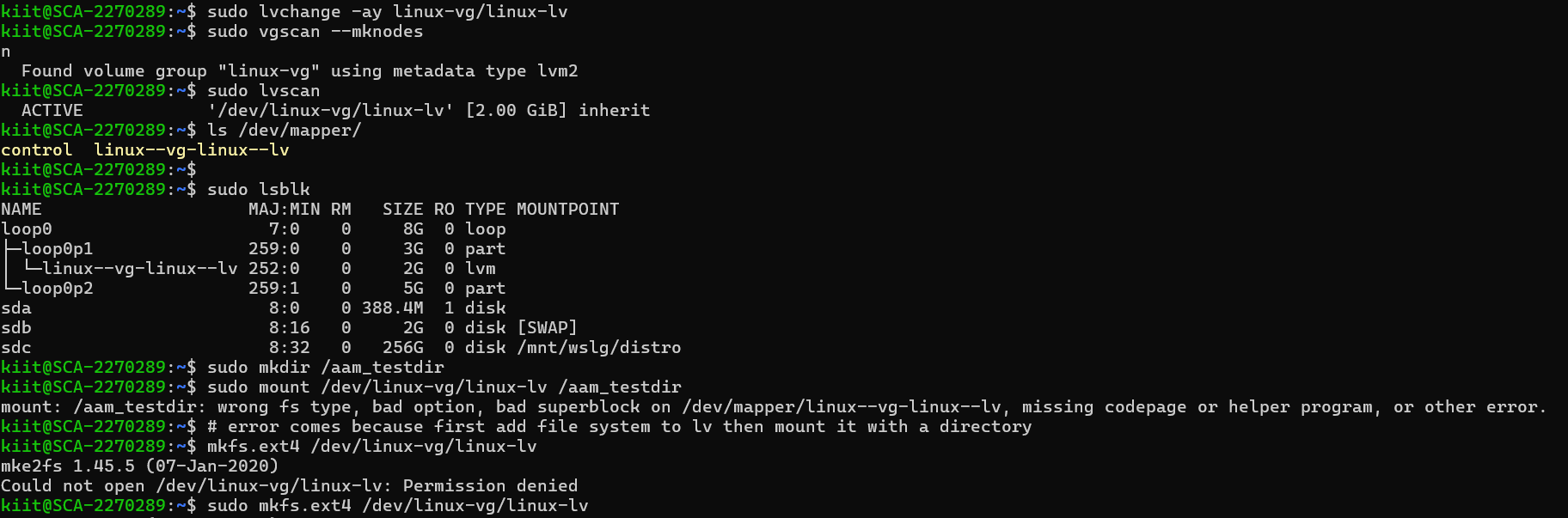
****

****

****

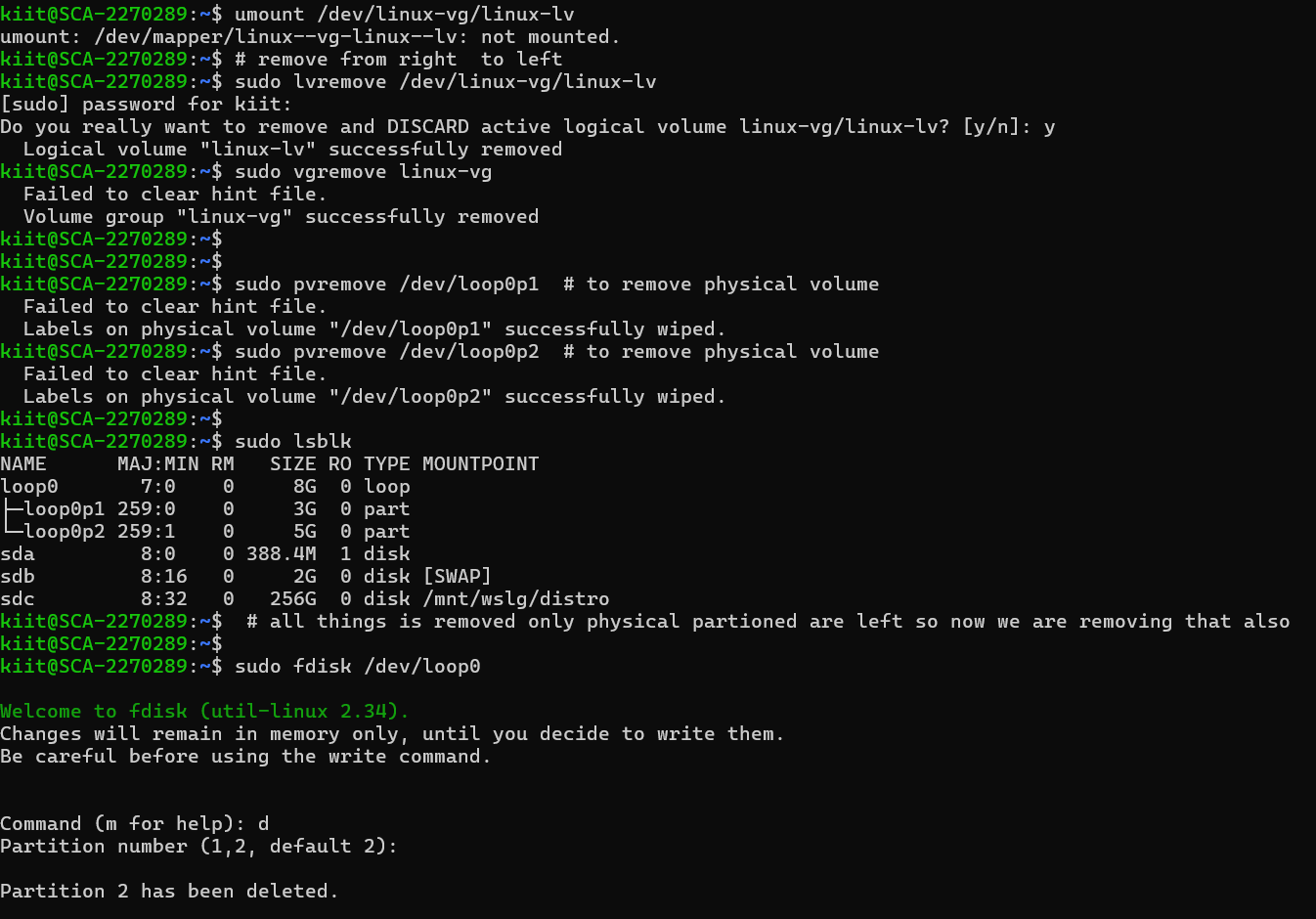


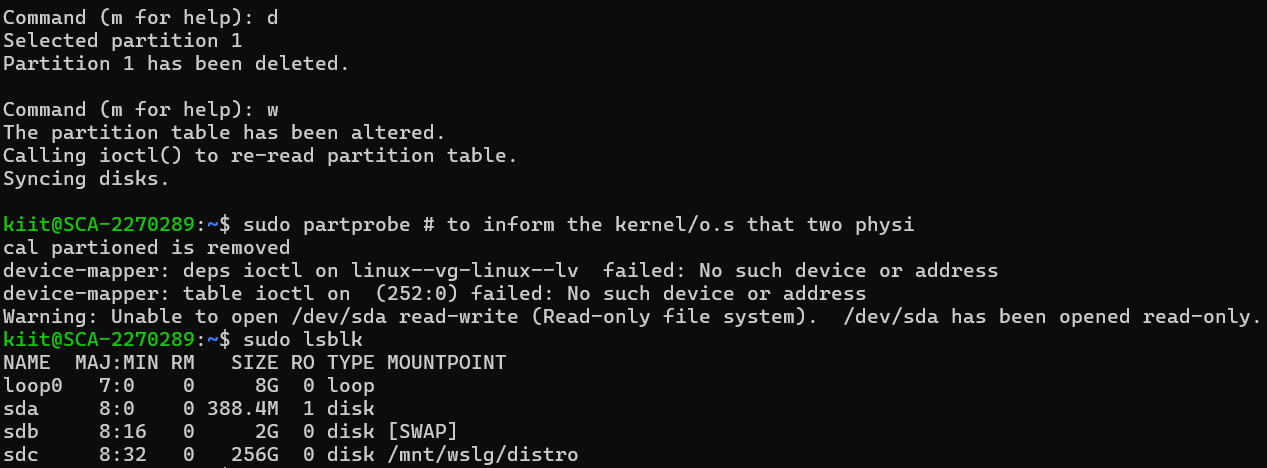




# 

## Below screenshots To remove/delete the things that I have done :





# Objective 6: Set up disk encryption using LUKS to secure sensitive data.

**Outcome:**

This lab will help you secure sensitive data by setting up disk encryption with LUKS, which is crucial for protecting data in transit or on disk.

1. Install Cryptsetup:

# Install cryptsetup to manage LUKS encryption.

2. Create an Encrypted Partition:

# Use cryptsetup luksFormat /dev/sdb1 to encrypt the partition.

3. Open Encrypted Volume:

# Use cryptsetup luksOpen /dev/sdb1 encrypted\_data to open the encrypted volume.

4. Create File System on Encrypted Partition:

# Format the opened volume with mkfs.ext4 or another file system.

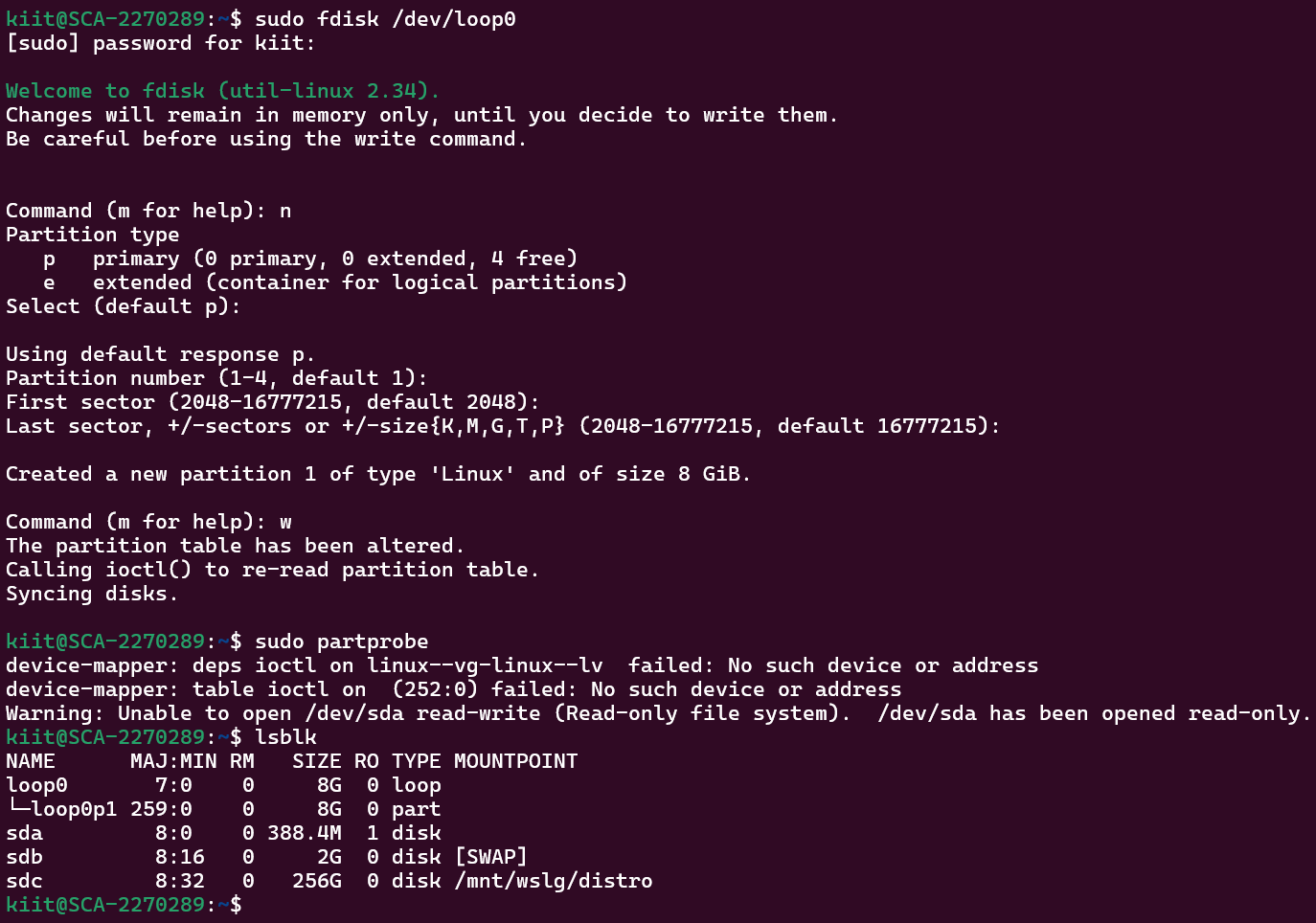
5. Mount and Configure Auto-Mount:

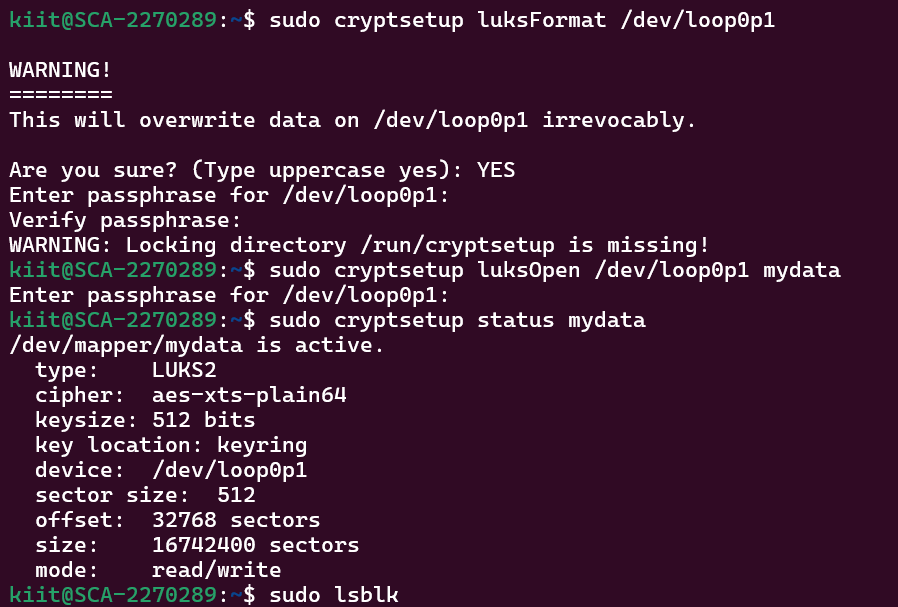
# Mount the encrypted partition and configure /etc/crypttab for automatic unlocking during boot.

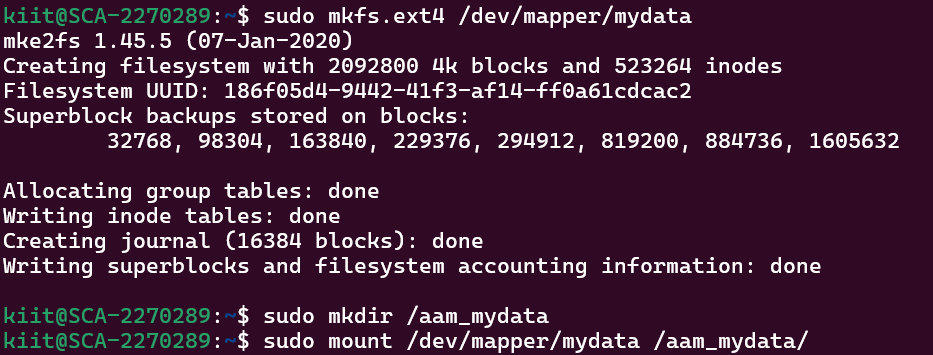
6. Verify Encryption:

# Test encryption by mounting the partition and ensuring data is unreadable without the correct passphrase.

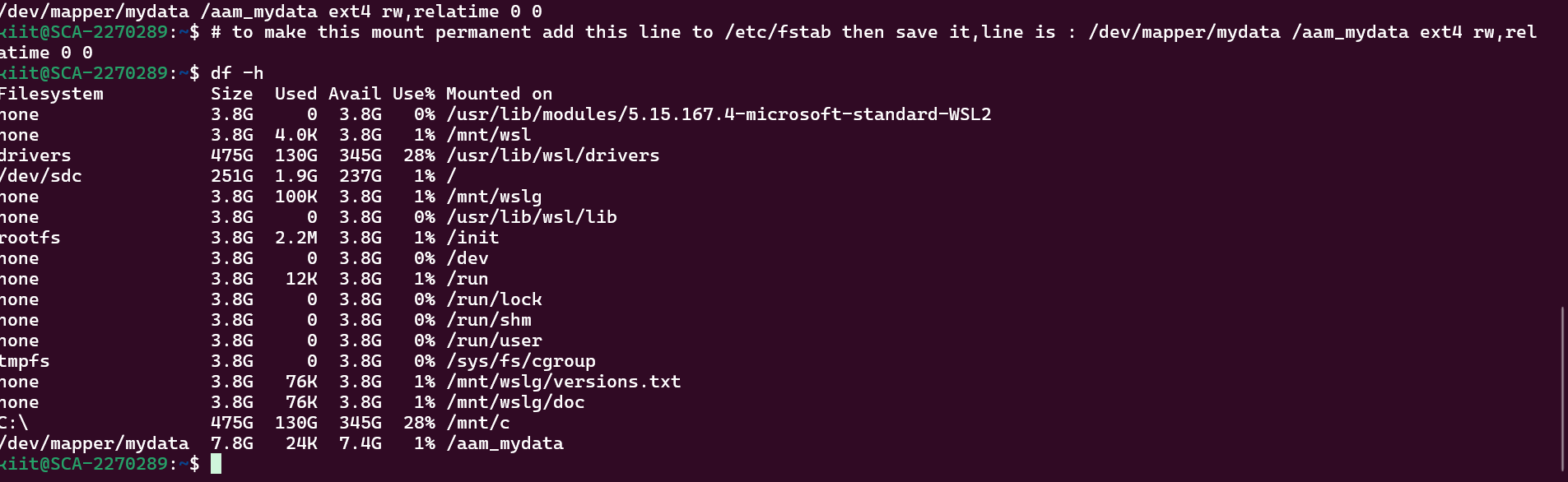
**Solution :**

****

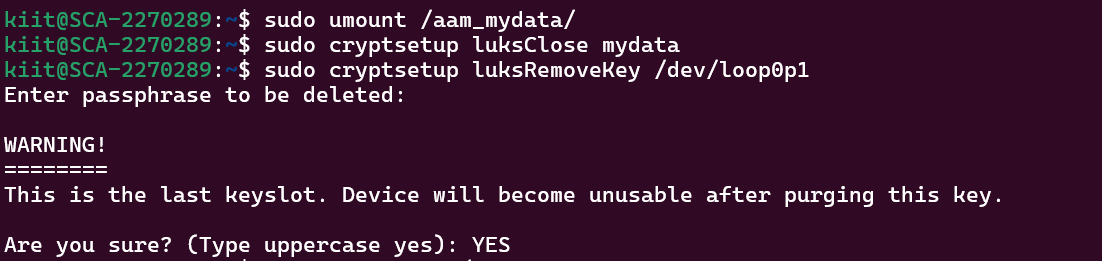
****

****

**cat /etc/mtab**

****

## To remove the process use this below screenshots.



**Objective 8 : File System Repair with fsck**

1. Simulate File System Corruption:

# Unmount a file system and use mount -o ro to create read-only access for a file system, simulating corruption.

2. Run fsck:

# Use fsck /dev/sdb1 to check and repair the file system.

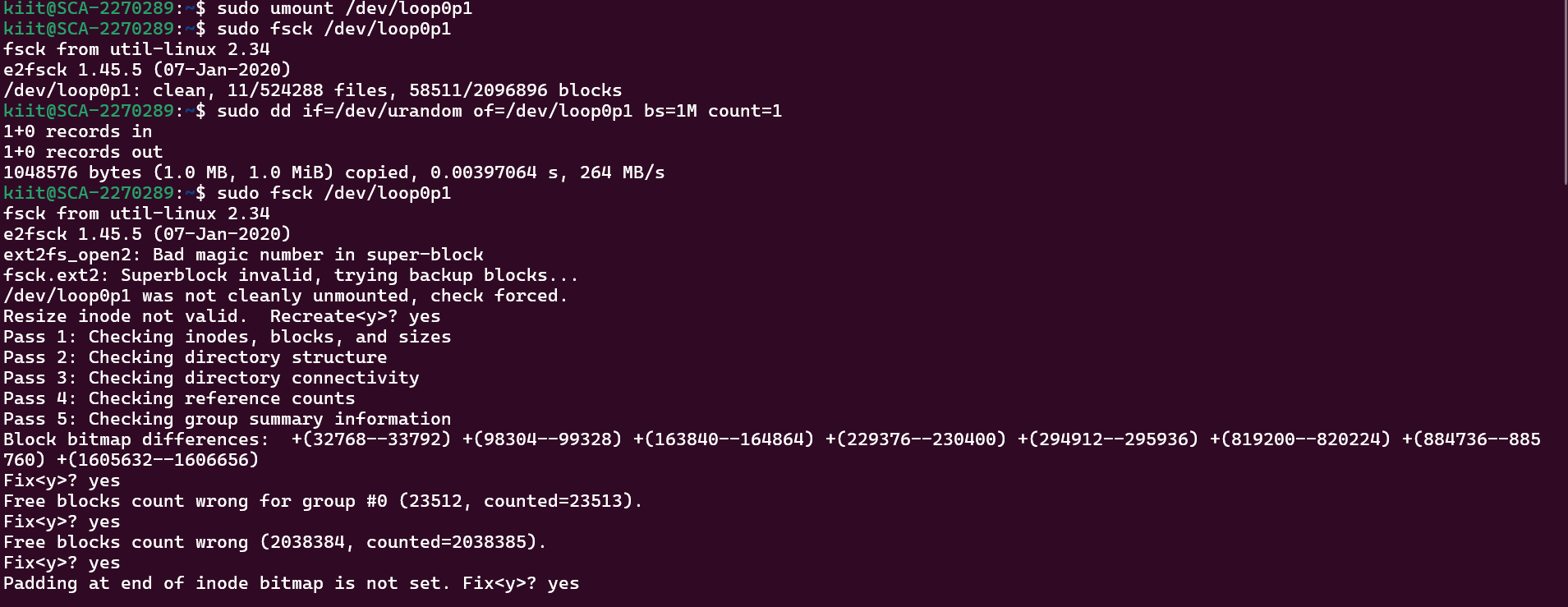
3. Repair Options:

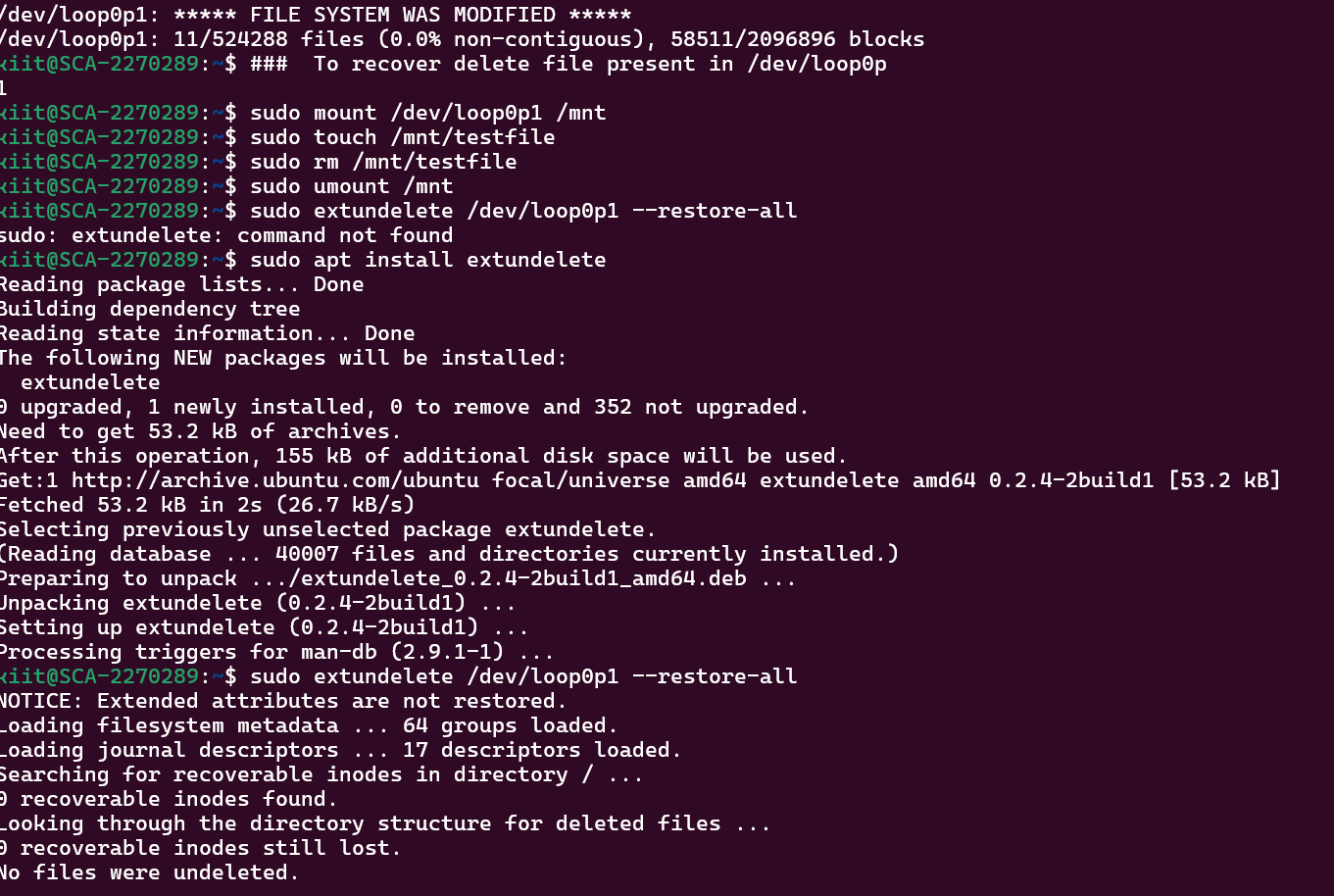
# Explore different fsck options such as -A (check all file systems) or -y (automatically fix errors).

4. Recover Lost Files:

# Use extundelete to attempt recovery of deleted files from an ext3/ext4 file system.

**Solution** :





# File System quotas

## Objective 9: Set up and manage file system quotas to control disk space usage for users and groups.

1. Enable Quotas on File System:

# Edit /etc/fstab to enable quotas on a partition (e.g., usrquota, grpquota).

# Remount the file system using mount -o remount /.

2. Create and Assign Quotas:

# Use edquota to set soft and hard disk quotas for users.

3. Monitor Quotas:

# Use repquota to generate reports on disk usage by users and groups.

4. Test Quotas:

# Test the quotas by trying to create files that exceed the assigned limits.

**Solution** :