Lab Project - 7

Objective: Linux filesystem management lab

DURATION: 2 - 3 Hourse

PRE-REQUISITES:

Oracle VirtualBox or VMWare, Ubuntu installed.

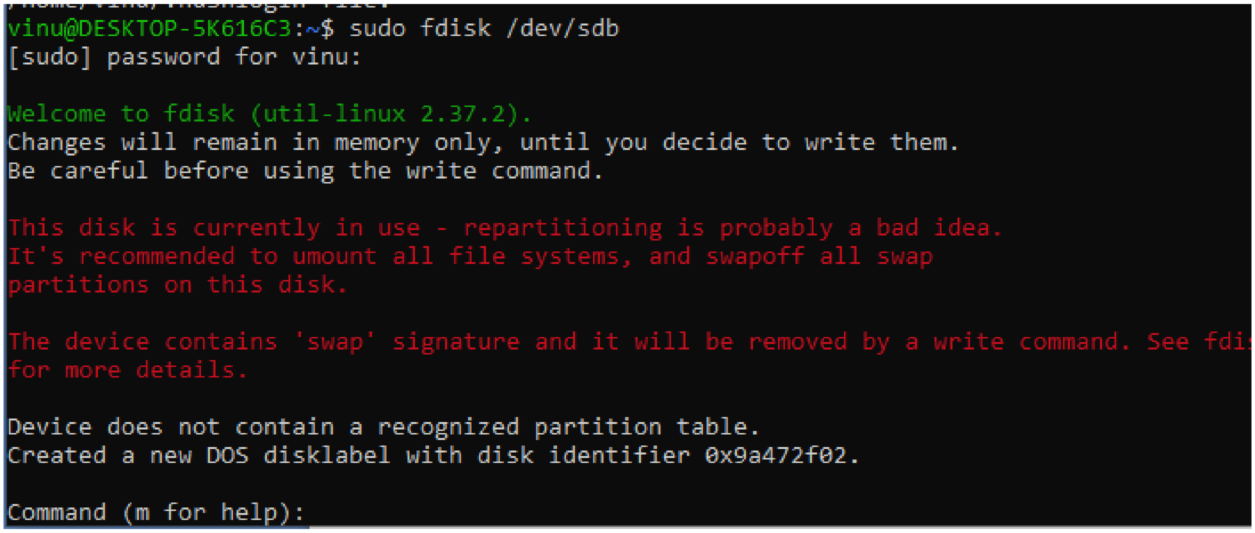
Lab 1: Disk Partitioning and File System Creation

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

Task:

```
 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.
```

Ans:-1



```
/home/vinu/.hushlogin file.
vinu@DESKTOP-5K616C3:~$ sudo fdisk /dev/sdb
[sudo] password for vinu:

Welcome to fdisk (util-linux 2.37.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

This disk is currently in use - repartitioning is probably a bad idea.
It's recommended to umount all file systems, and swapoff all swap
partitions on this disk.

The device contains 'swap' signature and it will be removed by a write command. See fdi
for more details.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0x9a472f02.

Command (m for help):
```

2. Create a primary partition

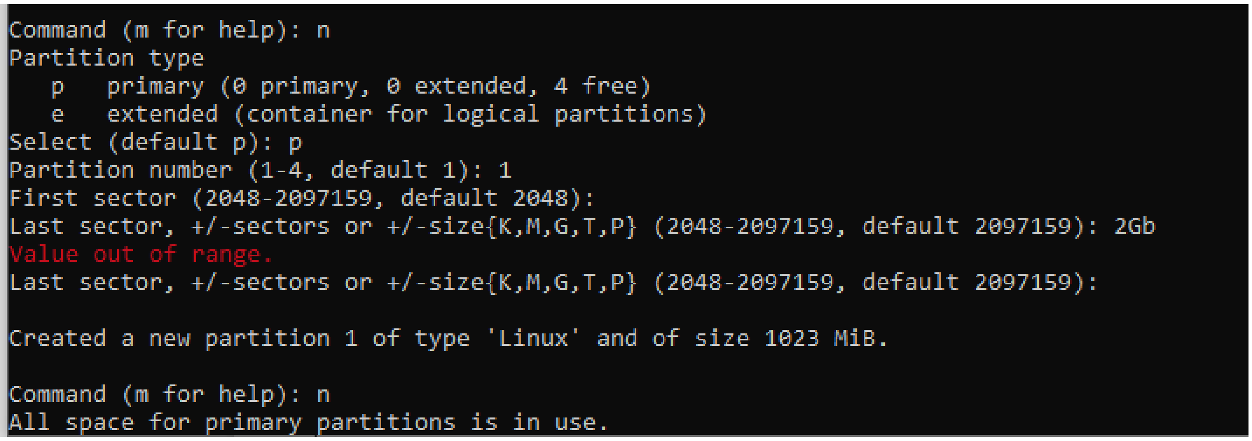
- Press `n` (new partition)
- Press `p` (primary partition)
- Select partition number (default: `1`)
- Accept the default first sector by pressing `Enter`
- Enter the size for the partition, e.g., `+10G` for 10GB

3. Create a swap partition

- Press `n` (new partition)
- Press `p` (primary partition) or `e` (extended for logical partitions)
- Select partition number (default: `2`)
- Accept default first sector
- Enter size for swap, e.g., `+2G` for 2GB
- Change partition type: Press `t` , select partition number, and enter `82` (Linux swap)

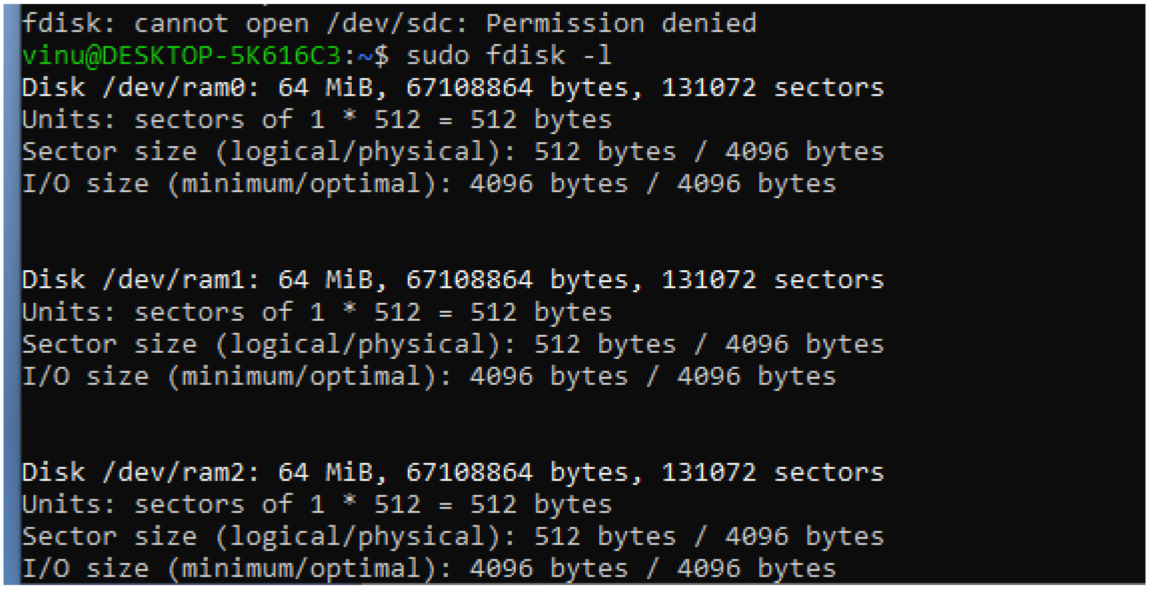
4. Write changes and exit

- Press `w` (write changes to disk)



```
Command (m for help): n
Partition type
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-2097159, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097159, default 2097159): 2Gb
Value out of range.
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-2097159, default 2097159):

Created a new partition 1 of type 'Linux' and of size 1023 MiB.

Command (m for help): n
All space for primary partitions is in use.
```

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



```
fdisk: cannot open /dev/sdc: Permission denied
vinu@DESKTOP-5K616C3:~$ sudo fdisk -l
Disk /dev/ram0: 64 MiB, 67108864 bytes, 131072 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/ram1: 64 MiB, 67108864 bytes, 131072 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes


Disk /dev/ram2: 64 MiB, 67108864 bytes, 131072 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
```

```
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.
```

1.1 Formatting a Partition as ext4

The `ext4` file system is a widely used journaling file system in Linux.

```bash
mkfs.ext4 /dev/sdX
```

For additional features:

```bash
mkfs.ext4 -L my_ext4 -m 1 -O ^has_journal /dev/sdX
```

- `-L my_ext4` : Sets the label of the partition.
- `-m 1` : Reserves 1% of space for root.
- `-O ^has_journal` : Disables journaling for performance.

1.3 Formatting a Partition as Btrfs

Btrfs provides snapshots and advanced features.

bash

```
mkfs.btrfs /dev/sdX
```

To create a multi-device Btrfs filesystem:

bash

```
mkfs.btrfs -L my_btrfs /dev/sdX /dev/sdY
```

1.2 Formatting a Partition as XFS

XFS is known for high performance, particularly in large file systems.

bash

```
mkfs.xfs /dev/sdX
```

To label it:

bash

```
mkfs.xfs -L my_xfs /dev/sdX
```

2. Checking File System Integrity with fsck

The `fsck` (File System Consistency Check) tool checks and repairs file system

⚠ **Important:** Unmount the partition before running `fsck`.

bash

```
umount /dev/sdX fsck /dev/sdX
```

2.1 Checking an ext4 File System

bash

```
fsck.ext4 -f /dev/sdX
```

- `-f` : Force check even if the file system is clean.

2.2 Checking an XFS File System

XFS uses `xfs_repair` instead of `fsck`.

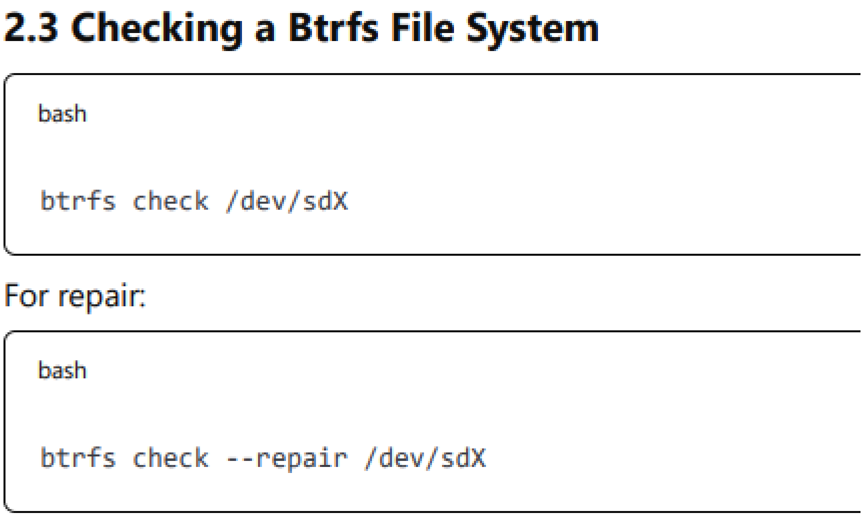
bash

```
xfs_repair /dev/sdX
```

If mounted:

bash

```
umount /dev/sdX && xfs_repair /dev/sdX
```



2.3 Checking a Btrfs File System

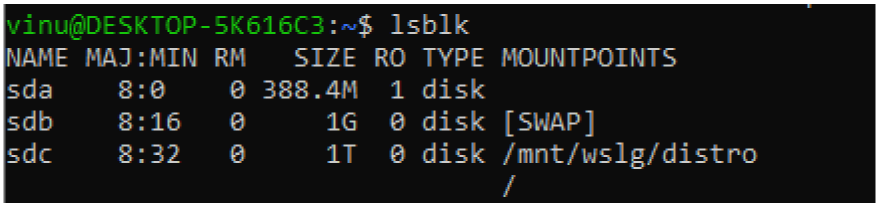
```bash
btrfs check /dev/sdX
```

For repair:

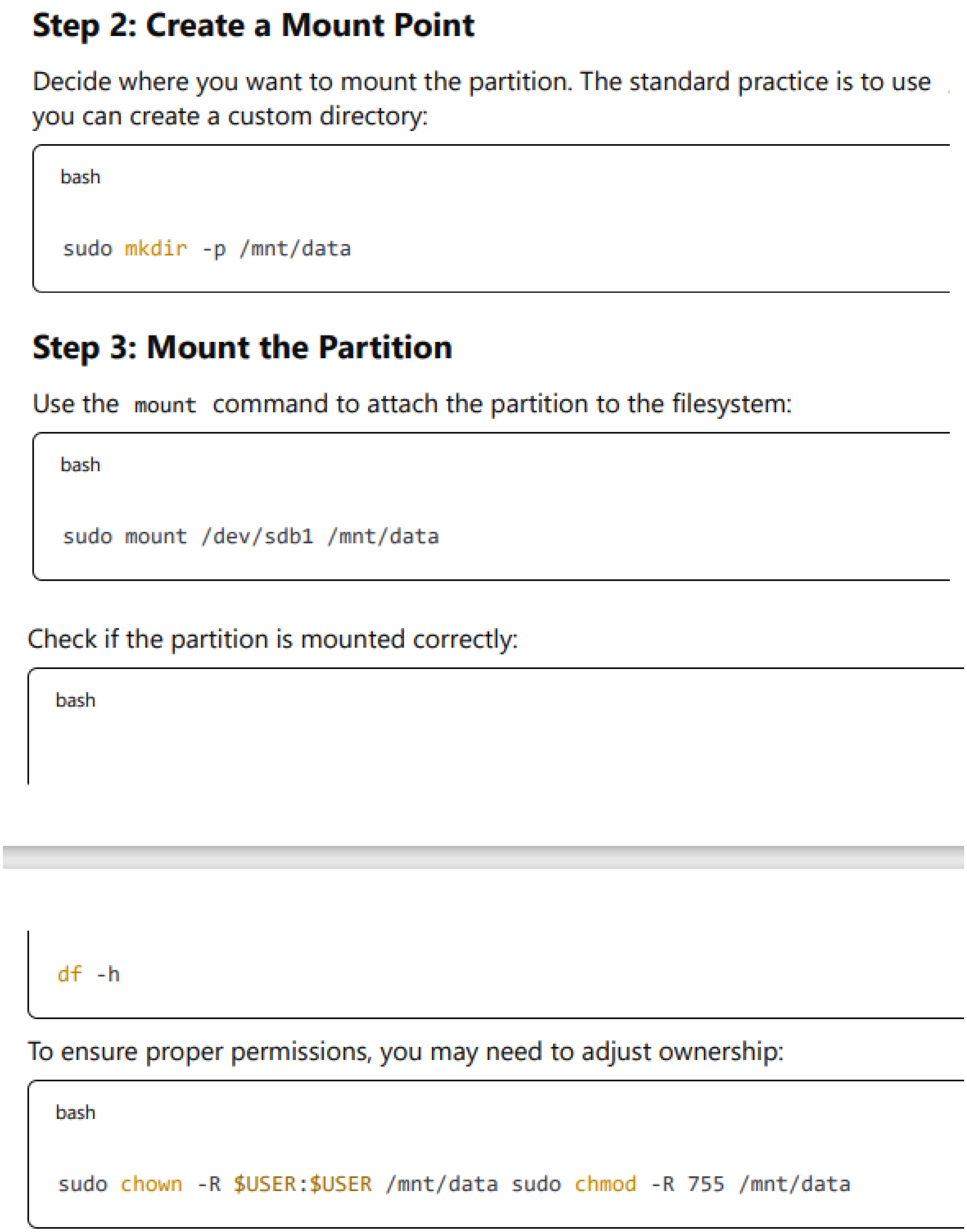
```bash
btrfs check --repair /dev/sdX
```

```
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.
```

Q:-a) Mount the new partitions manually using mount (e.g., mount /dev/sdb1 to /mnt/data).



```
vinu@DESKTOP-5K616C3:~$ lsblk
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 388.4M  1 disk
sdb    8:16   0     1G  0 disk [SWAP]
sdc    8:32   0     1T  0 disk /mnt/wslg/distro
                               /
```



Step 2: Create a Mount Point

Decide where you want to mount the partition. The standard practice is to use
you can create a custom directory:

bash

```bash
sudo mkdir -p /mnt/data
```

Step 3: Mount the Partition

Use the `mount` command to attach the partition to the filesystem:

bash

```bash
sudo mount /dev/sdb1 /mnt/data
```

Check if the partition is mounted correctly:

bash

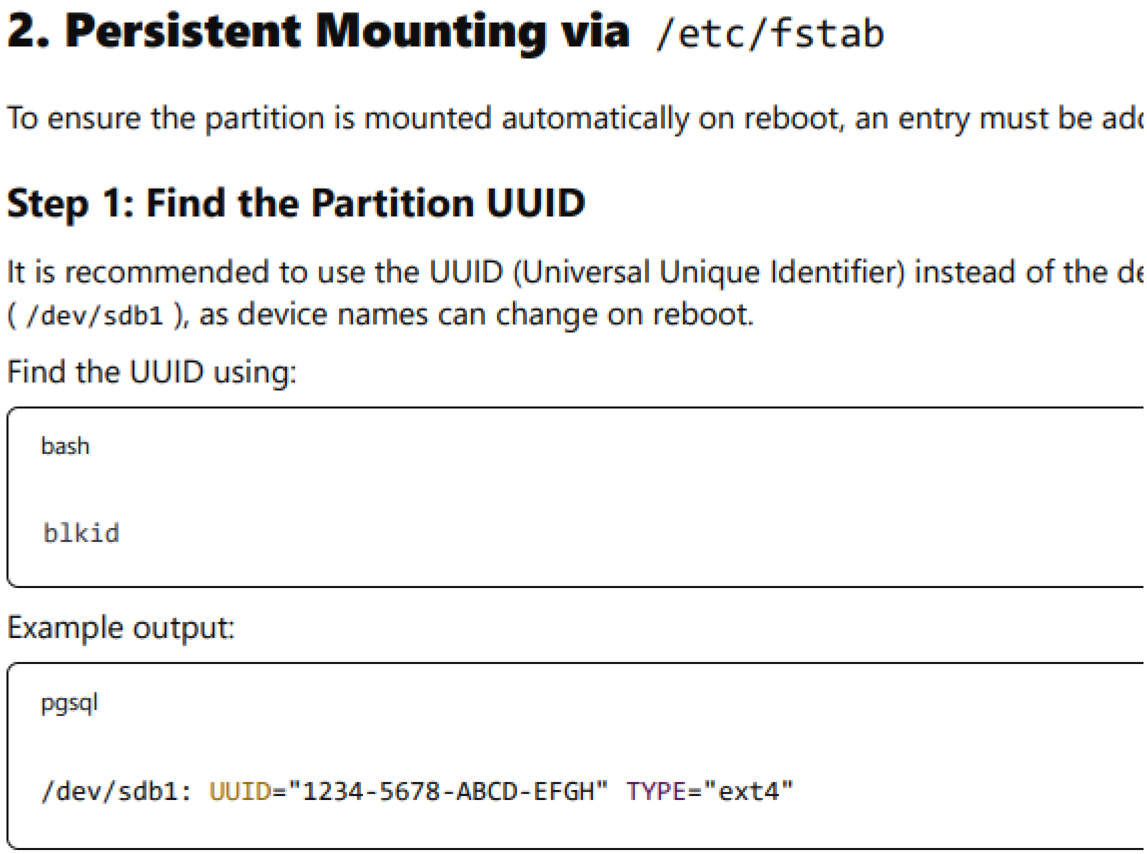
```bash
df -h
```

To ensure proper permissions, you may need to adjust ownership:

bash

```bash
sudo chown -R $USER:$USER /mnt/data sudo chmod -R 755 /mnt/data
```

Q:b) Add entries to /etc/fstab to ensure automatic mounting on boot.



2. Persistent Mounting via `/etc/fstab`

To ensure the partition is mounted automatically on reboot, an entry must be ad

Step 1: Find the Partition UUID

It is recommended to use the UUID (Universal Unique Identifier) instead of the de (`/dev/sdb1`), as device names can change on reboot.

Find the UUID using:

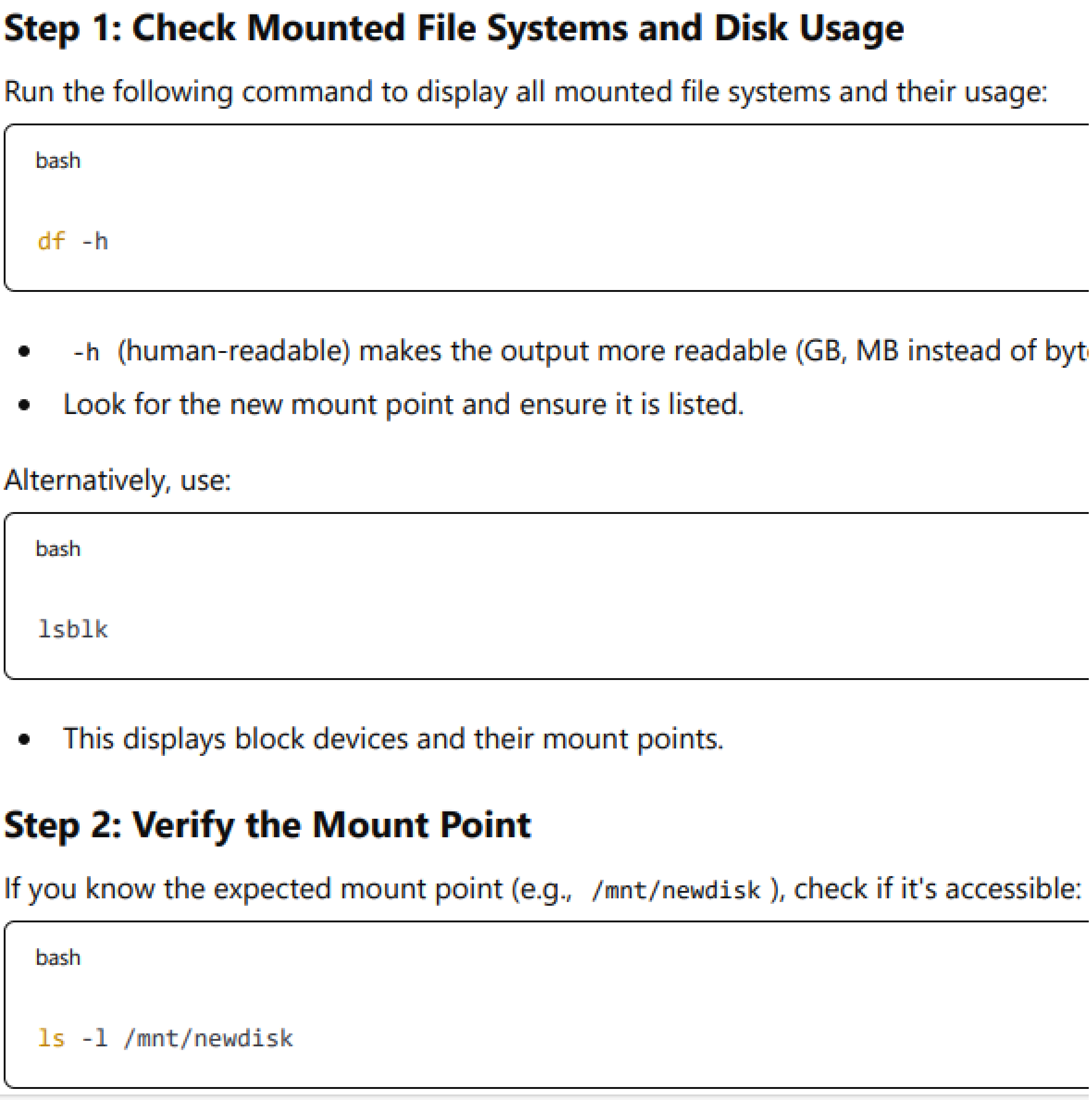
```bash
blkid
```

Example output:

```pgsql
/dev/sdb1: UUID="1234-5678-ABCD-EFGH" TYPE="ext4"
```

```
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.
```

Q:a)- Use df -h to check mounted file systems and disk usage.



Step 1: Check Mounted File Systems and Disk Usage

Run the following command to display all mounted file systems and their usage:

```bash
df -h
```

- `-h` (human-readable) makes the output more readable (GB, MB instead of byt
- Look for the new mount point and ensure it is listed.

Alternatively, use:

```bash
lsblk
```

- This displays block devices and their mount points.

Step 2: Verify the Mount Point

If you know the expected mount point (e.g., `/mnt/newdisk`), check if it's accessible:

```bash
ls -l /mnt/newdisk
```

Step 3: Test Read/Write Operations

Navigate to the new mount point:

bash

```bash
cd /mnt/newdisk
```

Write Test

Try creating a test file:

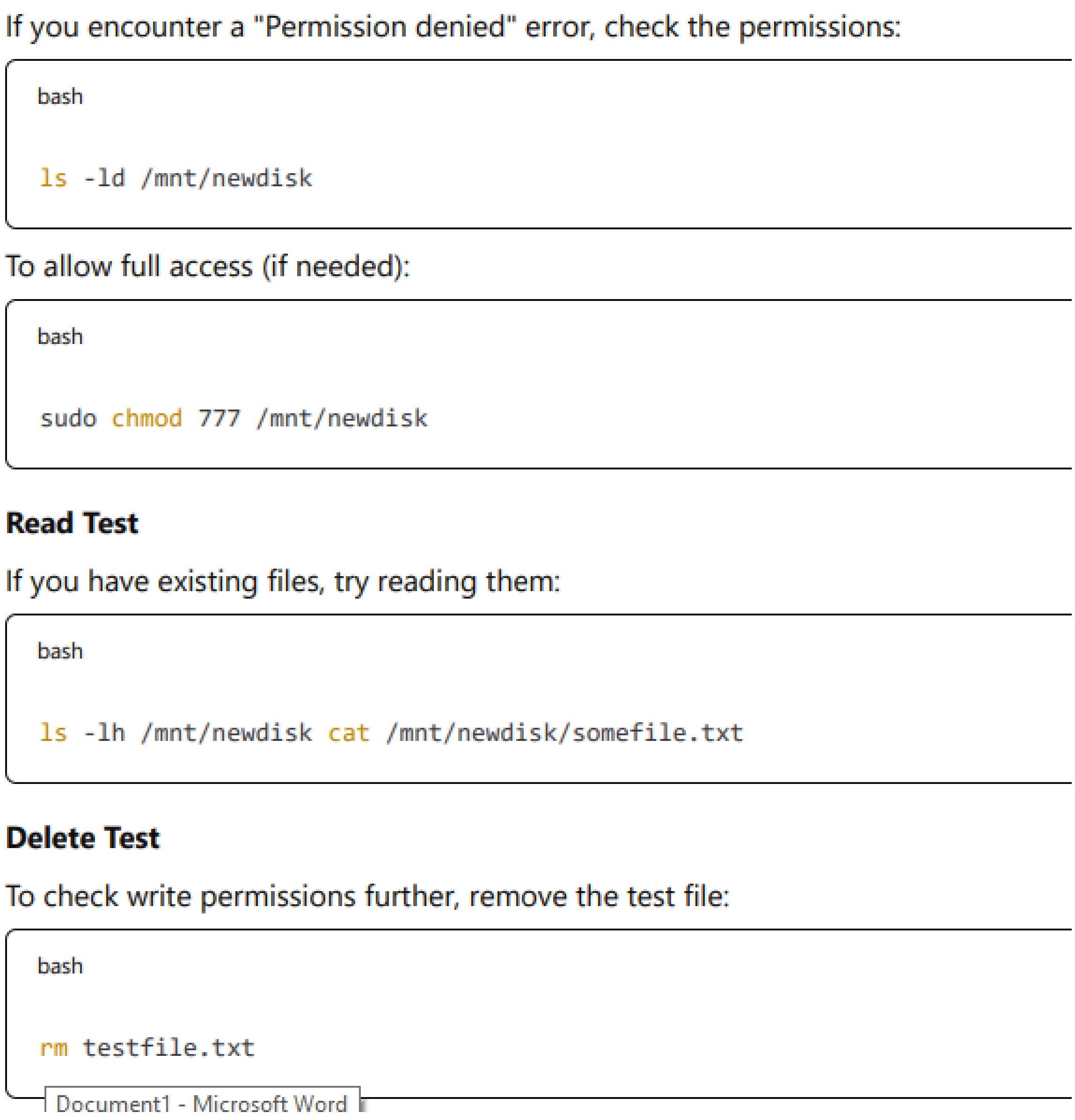
bash

```bash
echo "Mount verification test" > testfile.txt
```

Then, check if it was written:

bash

```bash
cat testfile.txt
```



If you encounter a "Permission denied" error, check the permissions:

bash

```bash
ls -ld /mnt/newdisk
```

To allow full access (if needed):

bash

```bash
sudo chmod 777 /mnt/newdisk
```

Read Test

If you have existing files, try reading them:

bash

```bash
ls -lh /mnt/newdisk cat /mnt/newdisk/somefile.txt
```

Delete Test

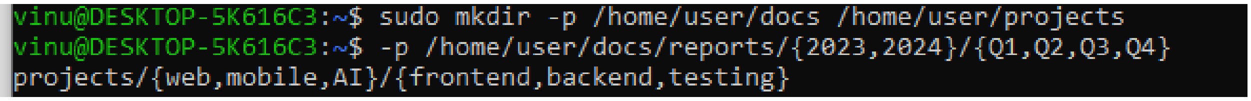
To check write permissions further, remove the test file:

bash

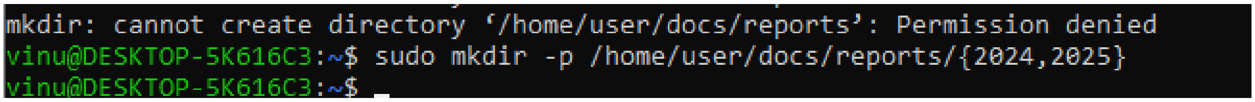
```bash
rm testfile.txt
```

Lab 2: Directory Structure and Permissions Management

```
1.     Create Directories:

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



```
vinu@DESKTOP-5K616C3:~$ sudo mkdir -p /home/user/docs /home/user/projects
vinu@DESKTOP-5K616C3:~$ -p /home/user/docs/reports/{2023,2024}/{Q1,Q2,Q3,Q4}
projects/{web,mobile,AI}/{frontend,backend,testing}
```



```
mkdir: cannot create directory ‘/home/user/docs/reports’: Permission denied
vinu@DESKTOP-5K616C3:~$ sudo mkdir -p /home/user/docs/reports/{2024,2025}
vinu@DESKTOP-5K616C3:~$
```

```
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.
```

1. File Permissions Overview

Each file and directory has three types of users:

- **Owner** (User): The person who created the file.
- **Group**: A set of users who share access.
- **Others**: Everyone else.

Permission Categories:

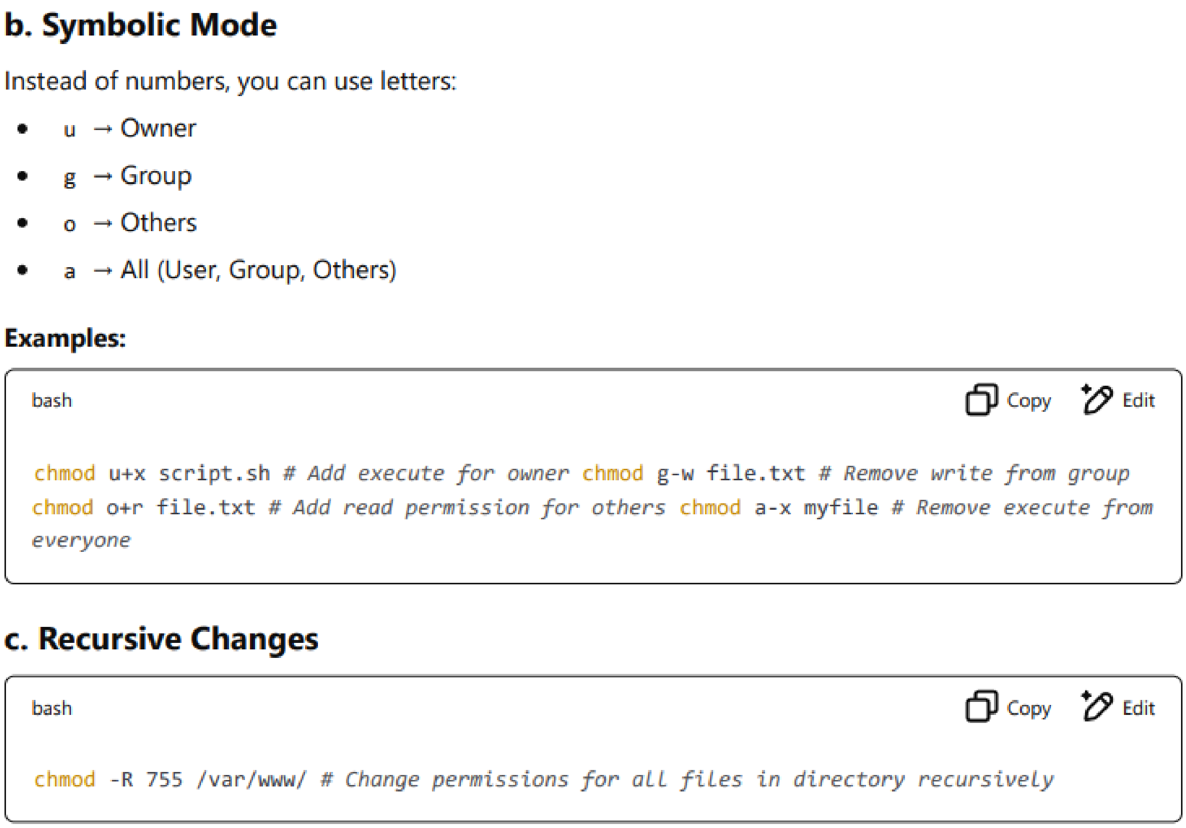
Symbol	Numeric	Description
`r` (read)	4	View the file contents
`w` (write)	2	Modify the file
`x` (execute)	1	Run the file (if executable)

Example of File Permissions:

bash

```
ls -l file.txt -rwxr-xr-- 1 user group 1024 Feb 18 10:00 file.txt
```

- `rwx` **(Owner)** → Read, Write, Execute
- `r-x` **(Group)** → Read, Execute
- `r--` **(Others)** → Read only



b. Symbolic Mode

Instead of numbers, you can use letters:

- u → Owner
- g → Group
- o → Others
- a → All (User, Group, Others)

Examples:

bash

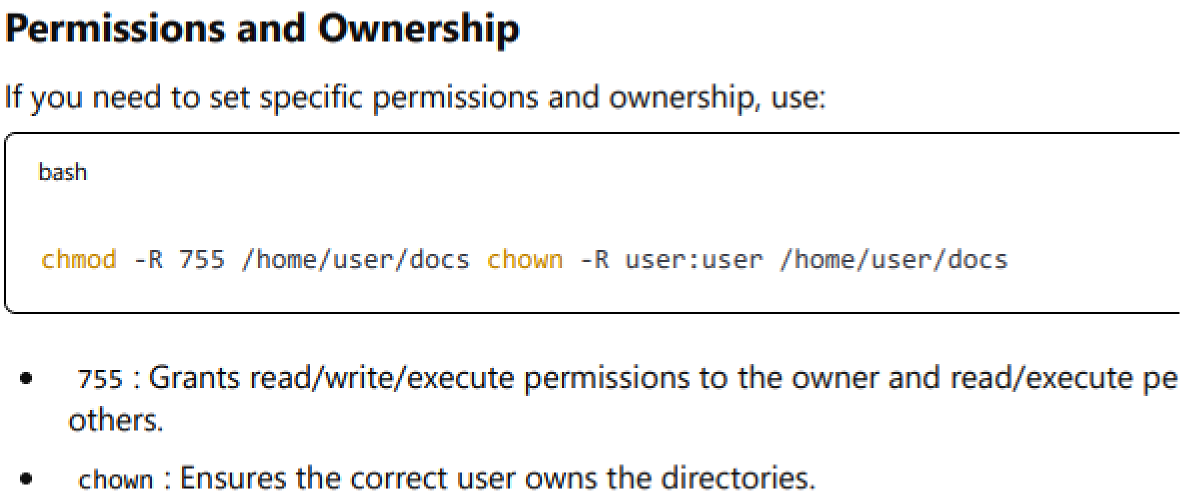
```bash
chmod u+x script.sh # Add execute for owner chmod g-w file.txt # Remove write from group
chmod o+r file.txt # Add read permission for others chmod a-x myfile # Remove execute from
everyone
```

c. Recursive Changes

bash

```bash
chmod -R 755 /var/www/ # Change permissions for all files in directory recursively
```

Q:b)- Use chown to change ownership of files and directories.



Permissions and Ownership

If you need to set specific permissions and ownership, use:

bash

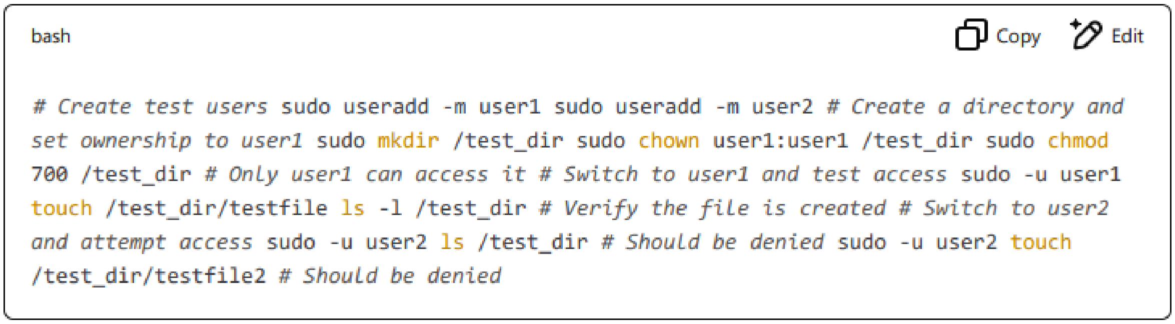
```bash
chmod -R 755 /home/user/docs chown -R user:user /home/user/docs
```

- 755 : Grants read/write/execute permissions to the owner and read/execute pe
others.
- chown : Ensures the correct user owns the directories.

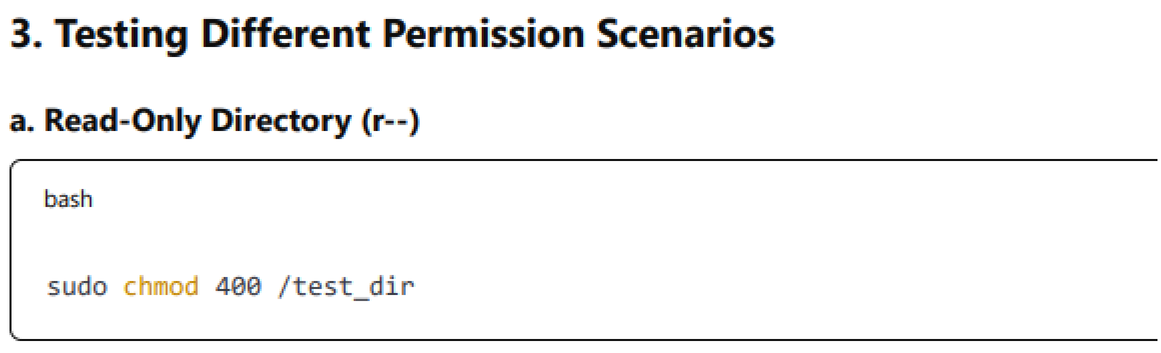
```
3.   Test Directory Permissions:

  #  Ensure that users without proper
permissions cannot access directories.
```

2. Creating a Test Environment

Run the following as a privileged user to create test directories and users:



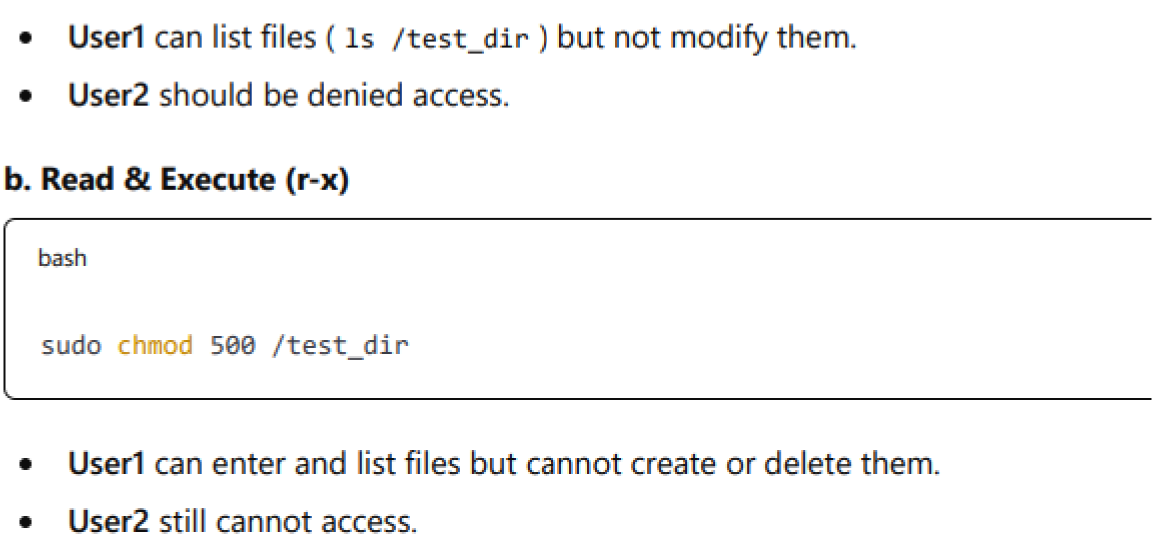
```bash
# Create test users sudo useradd -m user1 sudo useradd -m user2 # Create a directory and
set ownership to user1 sudo mkdir /test_dir sudo chown user1:user1 /test_dir sudo chmod
700 /test_dir # Only user1 can access it # Switch to user1 and test access sudo -u user1
touch /test_dir/testfile ls -l /test_dir # Verify the file is created # Switch to user2
and attempt access sudo -u user2 ls /test_dir # Should be denied sudo -u user2 touch
/test_dir/testfile2 # Should be denied
```



3. Testing Different Permission Scenarios

a. Read-Only Directory (r--)

```bash
sudo chmod 400 /test_dir
```



- **User1** can list files (`ls /test_dir`) but not modify them.
- **User2** should be denied access.

b. Read & Execute (r-x)

```bash
sudo chmod 500 /test_dir
```

- **User1** can enter and list files but cannot create or delete them.
- **User2** still cannot access.

c. Write-Only (w--)

```bash
sudo chmod 200 /test_dir
```

- **User1** can create/delete files but not list them.
- **User2** should be denied access.

d. Full Access for Group (rwxrwx---)

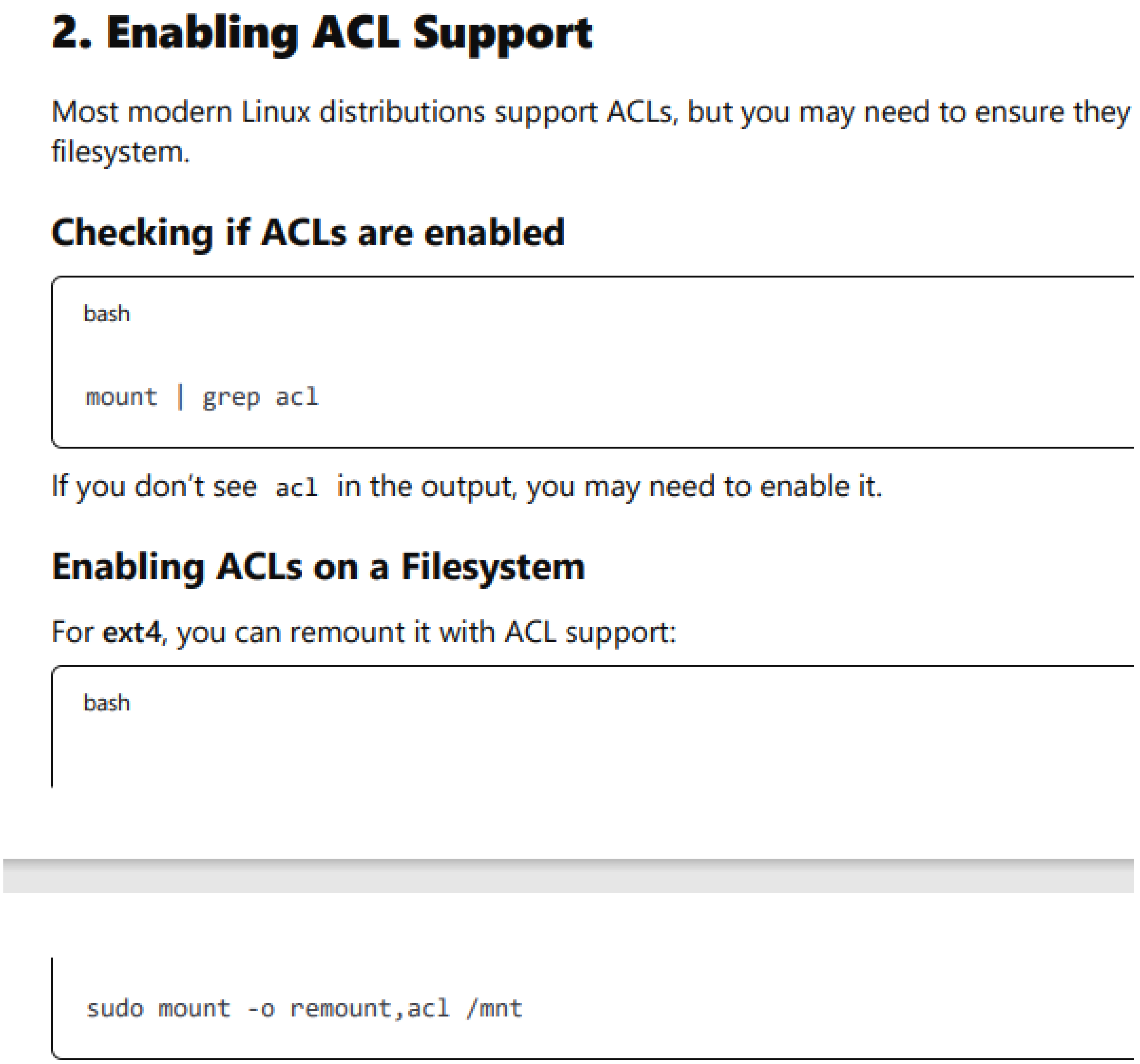
```bash
sudo chmod 770 /test_dir sudo chown user1:user2 /test_dir
```

- **User1 and User2** (since they are in the same group) can access the directory.
- **Others** are denied.

e. Public Access (rwxrwxrwx)

```bash
sudo chmod 777 /test_dir
```

```
4.      Use Access Control Lists (ACLs):

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



2. Enabling ACL Support

Most modern Linux distributions support ACLs, but you may need to ensure they filesystem.

Checking if ACLs are enabled

```bash
mount | grep acl
```

If you don't see `acl` in the output, you may need to enable it.

Enabling ACLs on a Filesystem

For **ext4**, you can remount it with ACL support:

```bash
sudo mount -o remount,acl /mnt
```

To make it permanent, add `acl` to `/etc/fstab`:

```bash
/dev/sdX    /mnt    ext4    defaults,acl    0  2
```

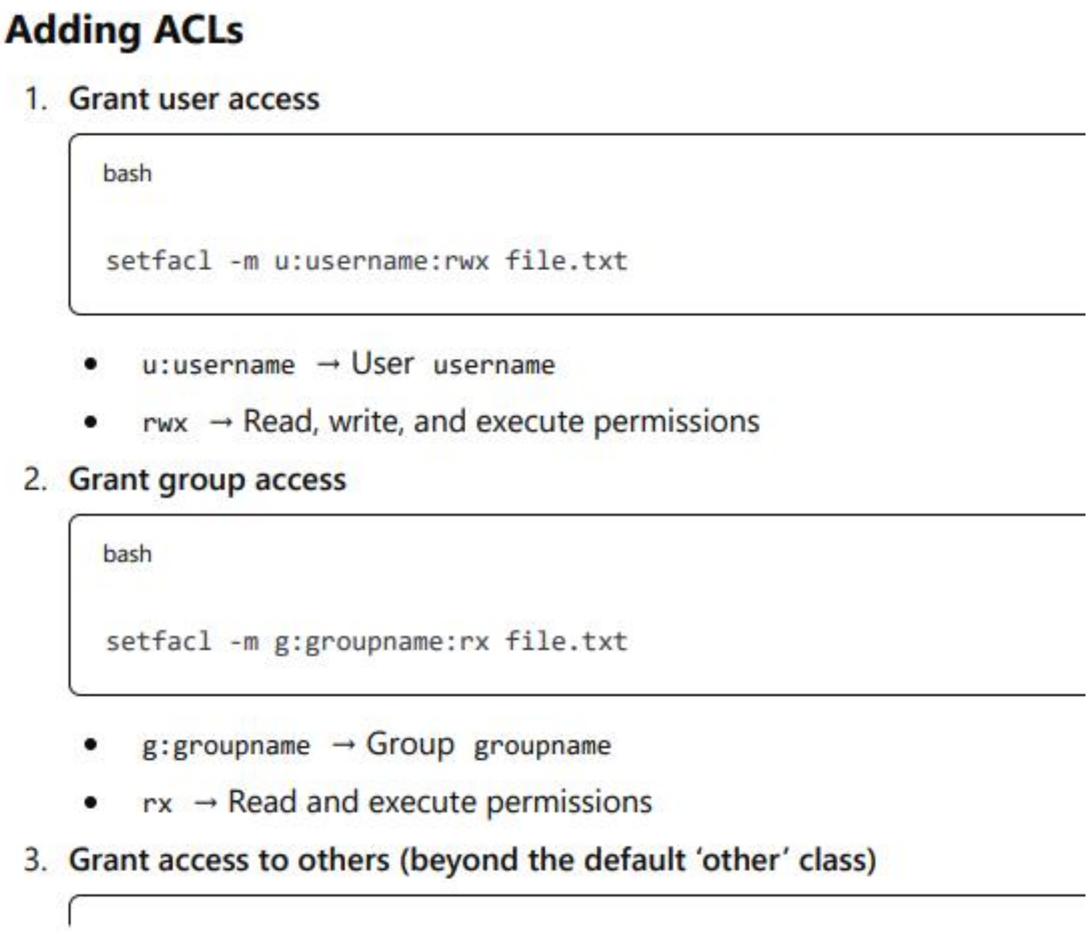
---

3. Using `setfacl` to Manage ACLs

The `setfacl` command is used to modify ACLs for files and directories.

Syntax

```bash
setfacl -m <permissions> <file/directory>
```



Adding ACLs

1. **Grant user access**

   bash

   ```
   setfacl -m u:username:rwx file.txt
   ```

   - `u:username` → User `username`
   - `rwx` → Read, write, and execute permissions

2. **Grant group access**

   bash

   ```
   setfacl -m g:groupname:rx file.txt
   ```

   - `g:groupname` → Group `groupname`
   - `rx` → Read and execute permissions

3. **Grant access to others (beyond the default ‘other’ class)**

```bash
setfacl -m o::r file.txt
```

- `o::r` → Others can only read

4. **Grant permissions recursively**

   ```bash
   setfacl -R -m u:username:rwx /path/to/directory
   ```

   - `-R` → Apply recursively to all files in the directory

---

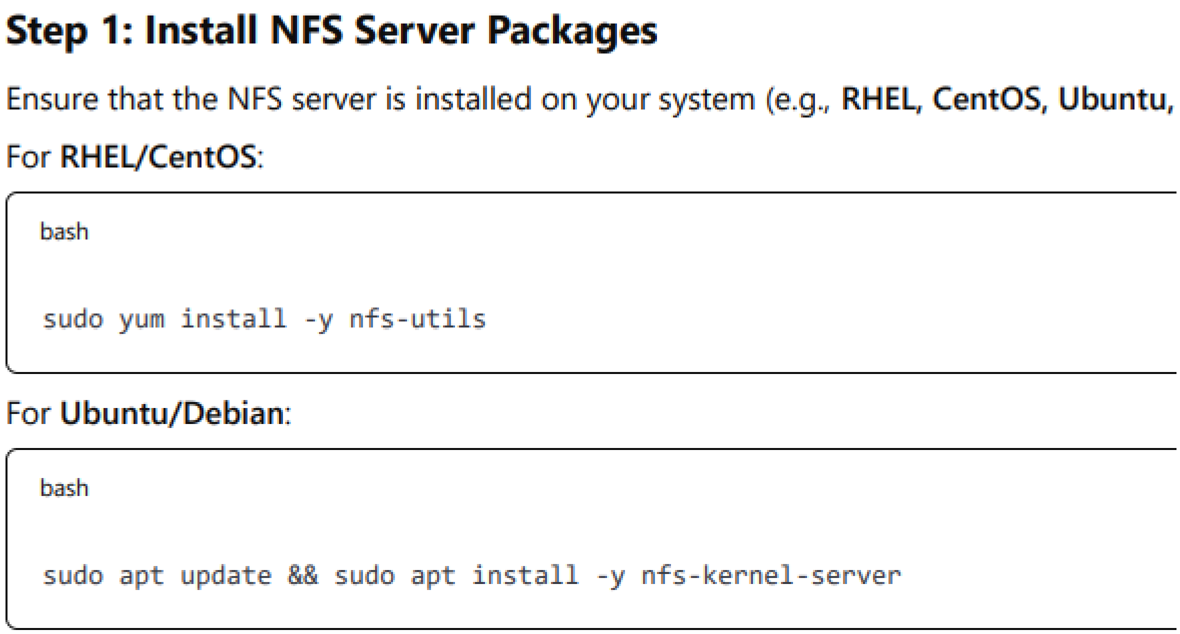
4. Checking ACLs

To view ACLs of a file:

```bash
getfacl file.txt
```

Lab 3: Mounting and Using Network File Systems (NFS)

```
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).
```



Step 1: Install NFS Server Packages

Ensure that the NFS server is installed on your system (e.g., **RHEL, CentOS, Ubuntu,**

For **RHEL/CentOS**:

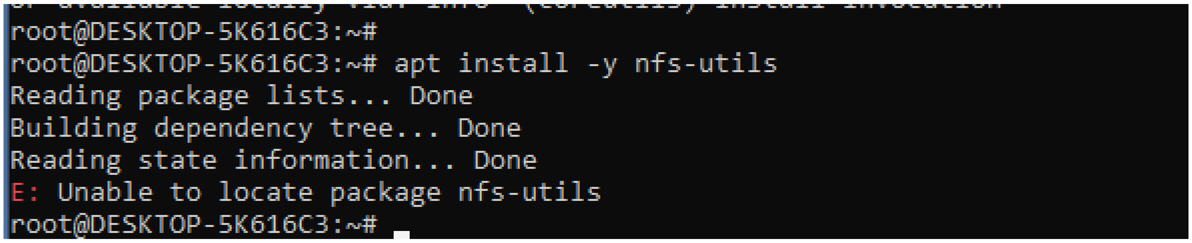
bash

```bash
sudo yum install -y nfs-utils
```

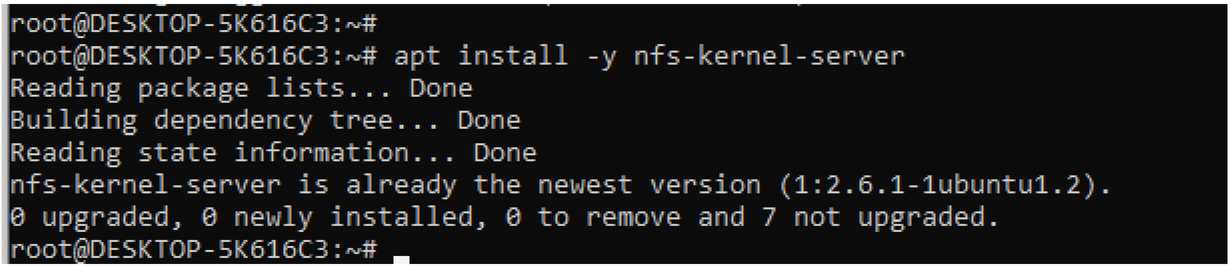
For **Ubuntu/Debian**:

bash

```bash
sudo apt update && sudo apt install -y nfs-kernel-server
```

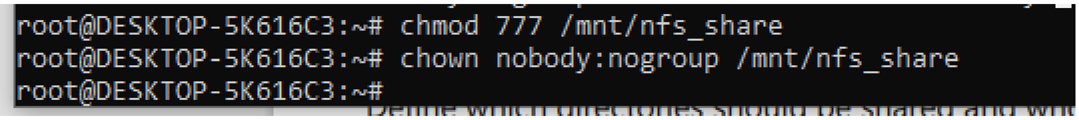


```
or available locally via: info '(coreutils) install invocation'
root@DESKTOP-5K616C3:~#
root@DESKTOP-5K616C3:~# apt install -y nfs-utils
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package nfs-utils
root@DESKTOP-5K616C3:~#
```

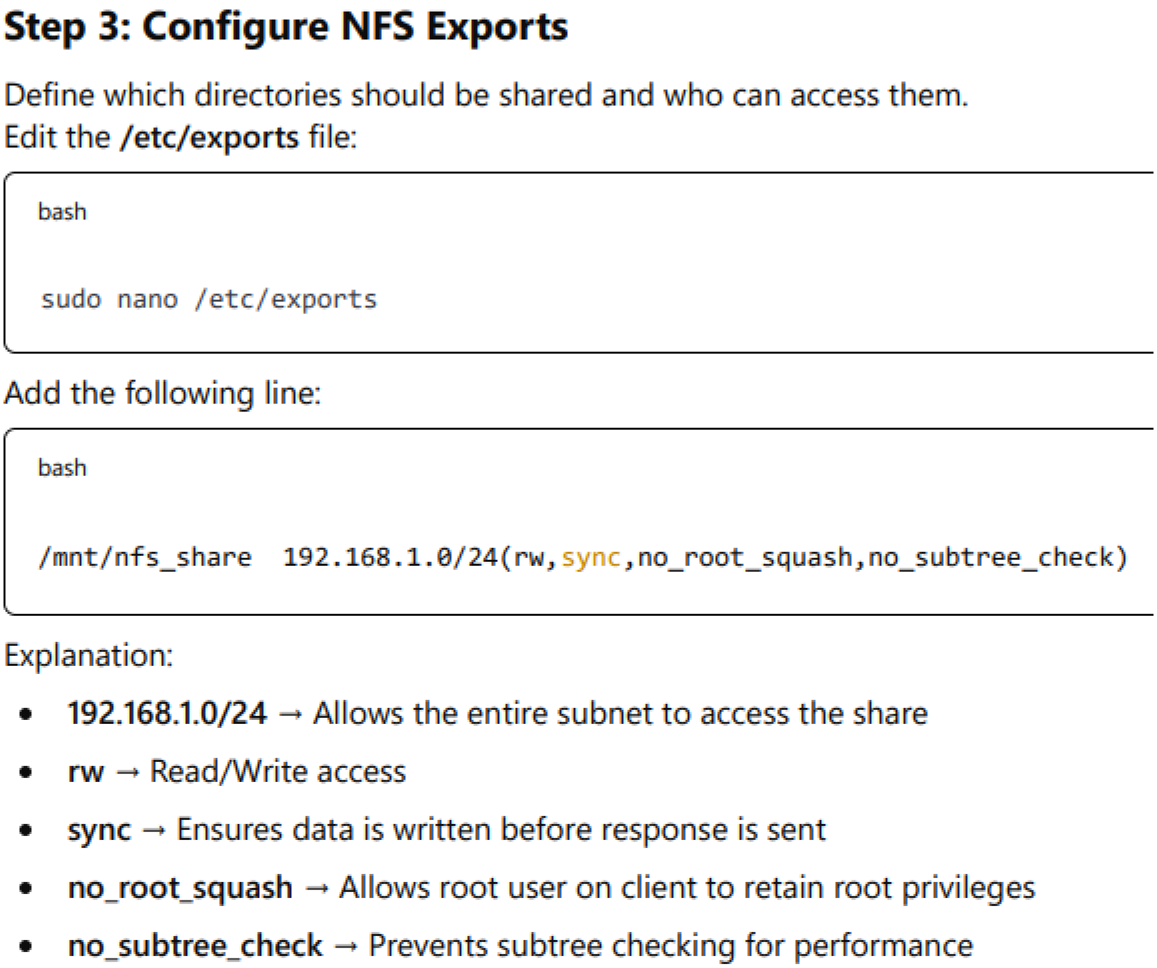


```
root@DESKTOP-5K616C3:~#
root@DESKTOP-5K616C3:~# apt install -y nfs-kernel-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
nfs-kernel-server is already the newest version (1:2.6.1-1ubuntu1.2).
0 upgraded, 0 newly installed, 0 to remove and 7 not upgraded.
root@DESKTOP-5K616C3:~#
```

Step 2: Create and Configure Shared Directory



```
root@DESKTOP-5K616C3:~# chmod 777 /mnt/nfs_share
root@DESKTOP-5K616C3:~# chown nobody:nogroup /mnt/nfs_share
root@DESKTOP-5K616C3:~#
```



Step 3: Configure NFS Exports

Define which directories should be shared and who can access them.
Edit the **/etc/exports** file:

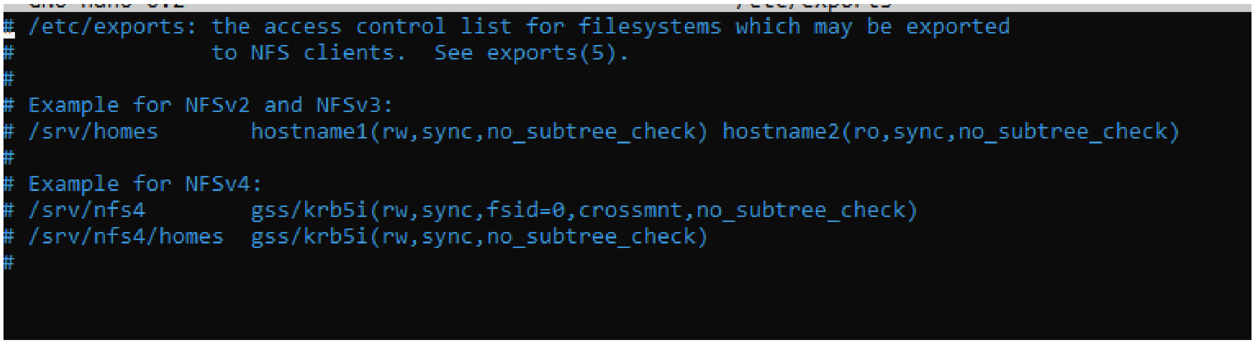
```bash
sudo nano /etc/exports
```

Add the following line:

```bash
/mnt/nfs_share  192.168.1.0/24(rw,sync,no_root_squash,no_subtree_check)
```

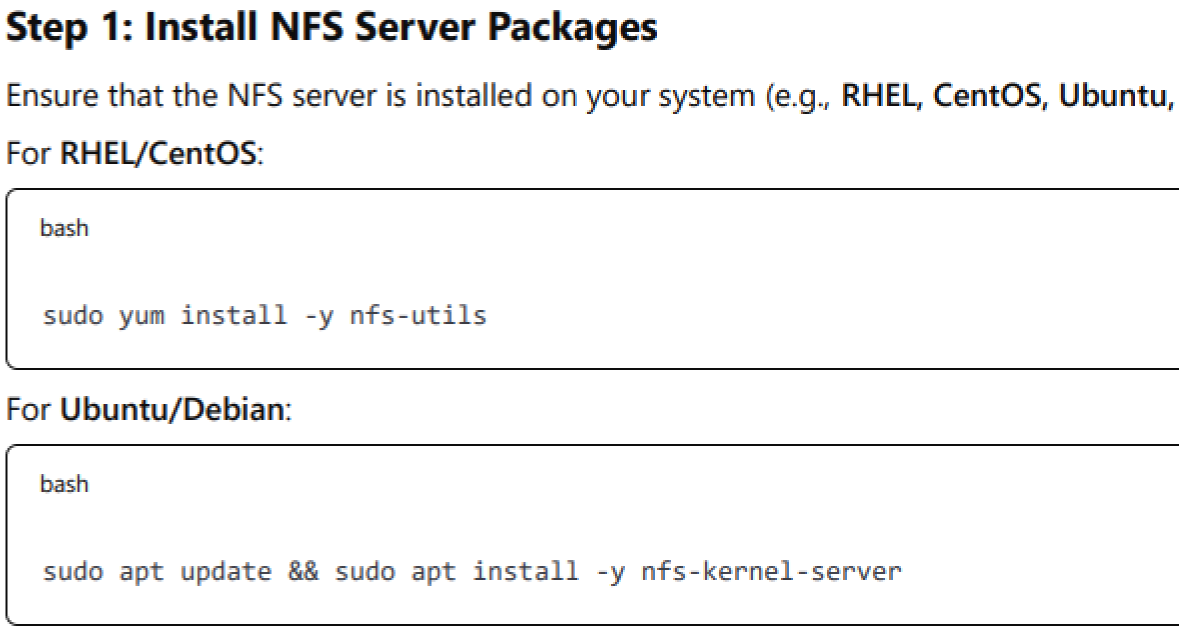
Explanation:

- **192.168.1.0/24** → Allows the entire subnet to access the share
- **rw** → Read/Write access
- **sync** → Ensures data is written before response is sent
- **no_root_squash** → Allows root user on client to retain root privileges
- **no_subtree_check** → Prevents subtree checking for performance



```
# /etc/exports: the access control list for filesystems which may be exported
#               to NFS clients.  See exports(5).
#
# Example for NFSv2 and NFSv3:
# /srv/homes       hostname1(rw,sync,no_subtree_check) hostname2(ro,sync,no_subtree_check)
#
# Example for NFSv4:
# /srv/nfs4        gss/krb5i(rw,sync,fsid=0,crossmnt,no_subtree_check)
# /srv/nfs4/homes  gss/krb5i(rw,sync,no_subtree_check)
#
```

```
2.      Configure NFS Server:
  #     Export the shared directory using the exportfs command.
  #     Start the NFS service with systemctl start nfs-server
```



Step 1: Install NFS Server Packages

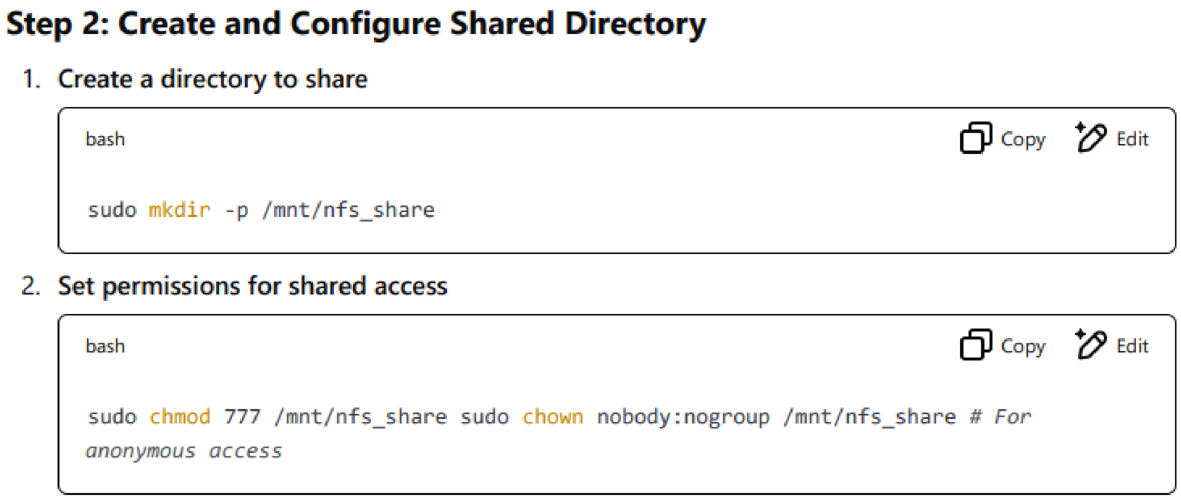
Ensure that the NFS server is installed on your system (e.g., **RHEL, CentOS, Ubuntu,**

For **RHEL/CentOS**:

```bash
sudo yum install -y nfs-utils
```

For **Ubuntu/Debian**:

```bash
sudo apt update && sudo apt install -y nfs-kernel-server
```



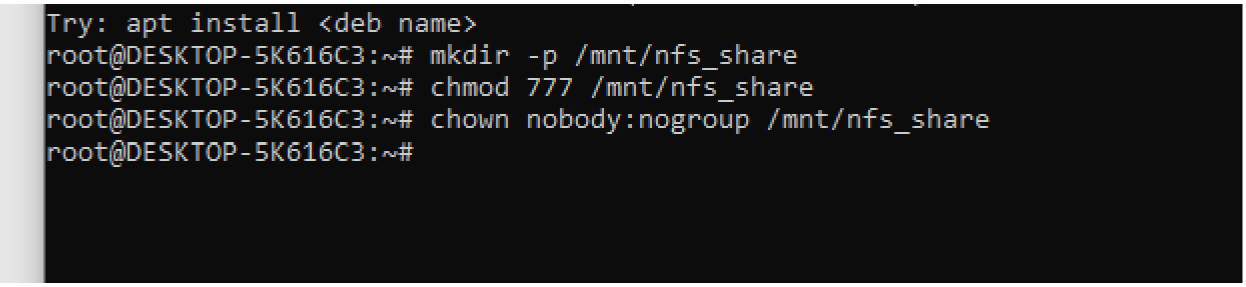
Step 2: Create and Configure Shared Directory

1. **Create a directory to share**

```bash
sudo mkdir -p /mnt/nfs_share
```

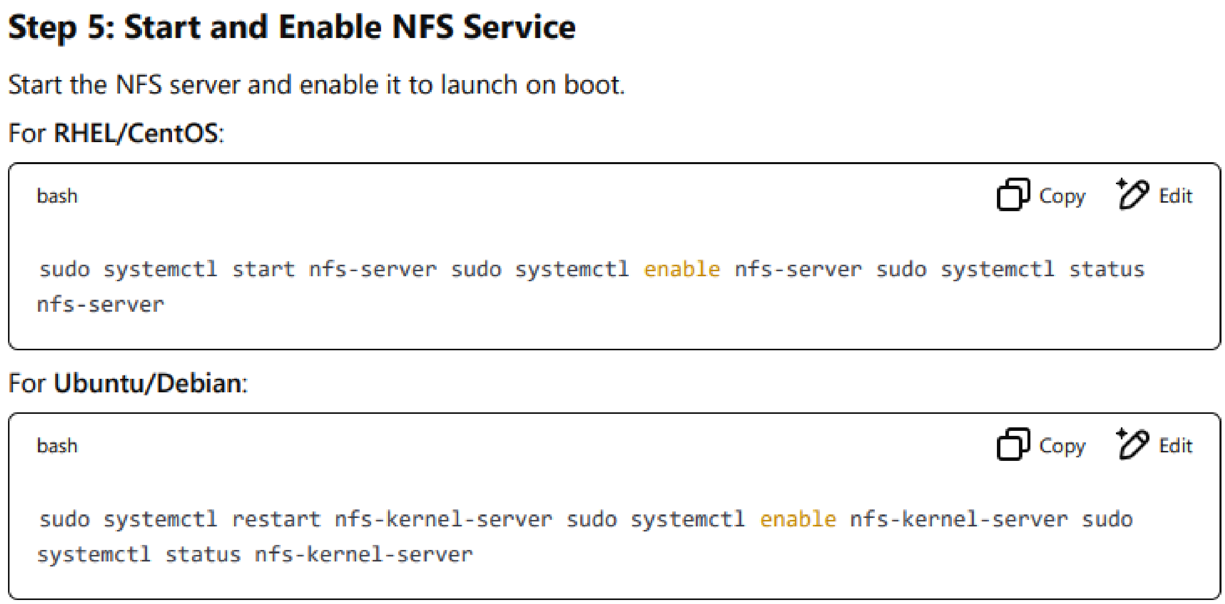
2. **Set permissions for shared access**

```bash
sudo chmod 777 /mnt/nfs_share sudo chown nobody:nogroup /mnt/nfs_share # For
anonymous access
```



```
Try: apt install <deb name>
root@DESKTOP-5K616C3:~# mkdir -p /mnt/nfs_share
root@DESKTOP-5K616C3:~# chmod 777 /mnt/nfs_share
root@DESKTOP-5K616C3:~# chown nobody:nogroup /mnt/nfs_share
root@DESKTOP-5K616C3:~#
```

Q:b)- Start the NFS service with systemctl start nfs-server



Step 5: Start and Enable NFS Service

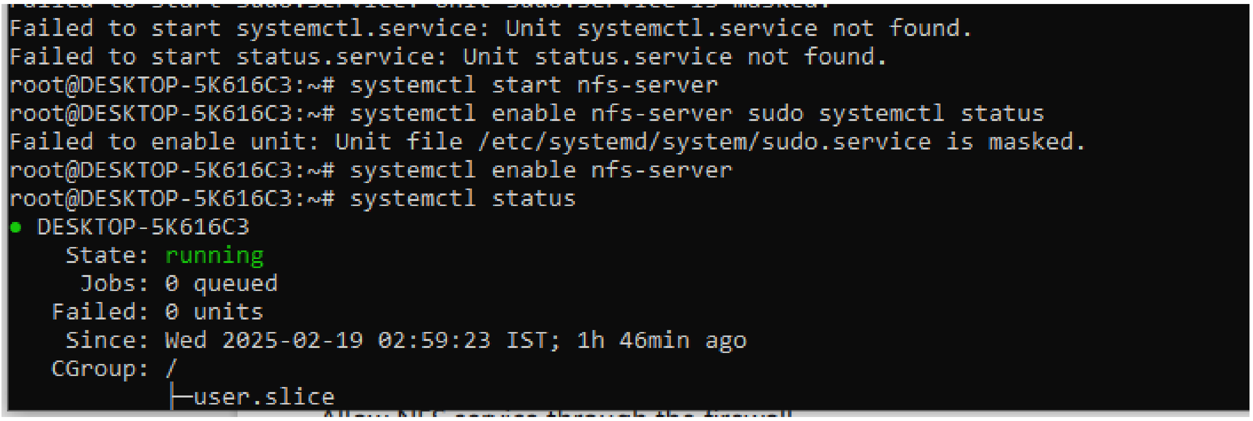
Start the NFS server and enable it to launch on boot.

For **RHEL/CentOS**:

```bash
sudo systemctl start nfs-server sudo systemctl enable nfs-server sudo systemctl status nfs-server
```

For **Ubuntu/Debian**:

```bash
sudo systemctl restart nfs-kernel-server sudo systemctl enable nfs-kernel-server sudo systemctl status nfs-kernel-server
```



```
Failed to start sudo.service: Unit sudo.service is masked.
Failed to start systemctl.service: Unit systemctl.service not found.
Failed to start status.service: Unit status.service not found.
root@DESKTOP-5K616C3:~# systemctl start nfs-server
root@DESKTOP-5K616C3:~# systemctl enable nfs-server sudo systemctl status
Failed to enable unit: Unit file /etc/systemd/system/sudo.service is masked.
root@DESKTOP-5K616C3:~# systemctl enable nfs-server
root@DESKTOP-5K616C3:~# systemctl status
● DESKTOP-5K616C3
    State: running
     Jobs: 0 queued
   Failed: 0 units
    Since: Wed 2025-02-19 02:59:23 IST; 1h 46min ago
   CGroup: /
           ├─user.slice
```

```
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).
```

Check NFS Service Status

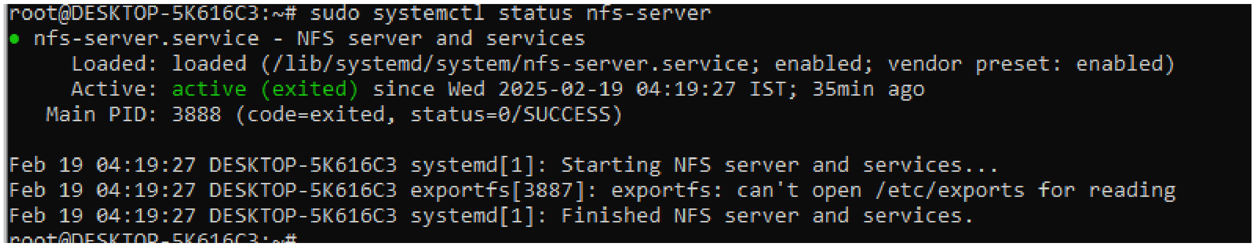
```bash
sudo systemctl status nfs-server
```

Restart NFS Service

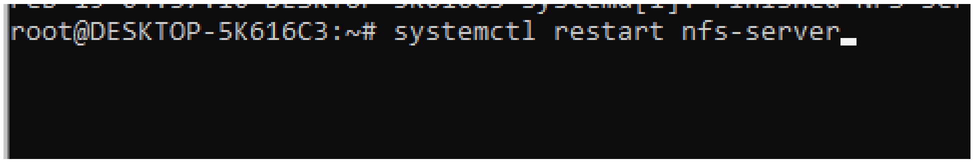
```bash
sudo systemctl restart nfs-server
```

Verify Shared Directories

```bash
exportfs -v
```



```
root@DESKTOP-5K616C3:~# sudo systemctl status nfs-server
● nfs-server.service - NFS server and services
     Loaded: loaded (/lib/systemd/system/nfs-server.service; enabled; vendor preset: enabled)
     Active: active (exited) since Wed 2025-02-19 04:19:27 IST; 35min ago
   Main PID: 3888 (code=exited, status=0/SUCCESS)

Feb 19 04:19:27 DESKTOP-5K616C3 systemd[1]: Starting NFS server and services...
Feb 19 04:19:27 DESKTOP-5K616C3 exportfs[3887]: exportfs: can't open /etc/exports for reading
Feb 19 04:19:27 DESKTOP-5K616C3 systemd[1]: Finished NFS server and services.
root@DESKTOP-5K616C3:~#
```



```
Feb 19 04:19:27 DESKTOP-5K616C3 systemd[1]: Finished NFS ser
root@DESKTOP-5K616C3:~# systemctl restart nfs-server
```

Check Network Connectivity

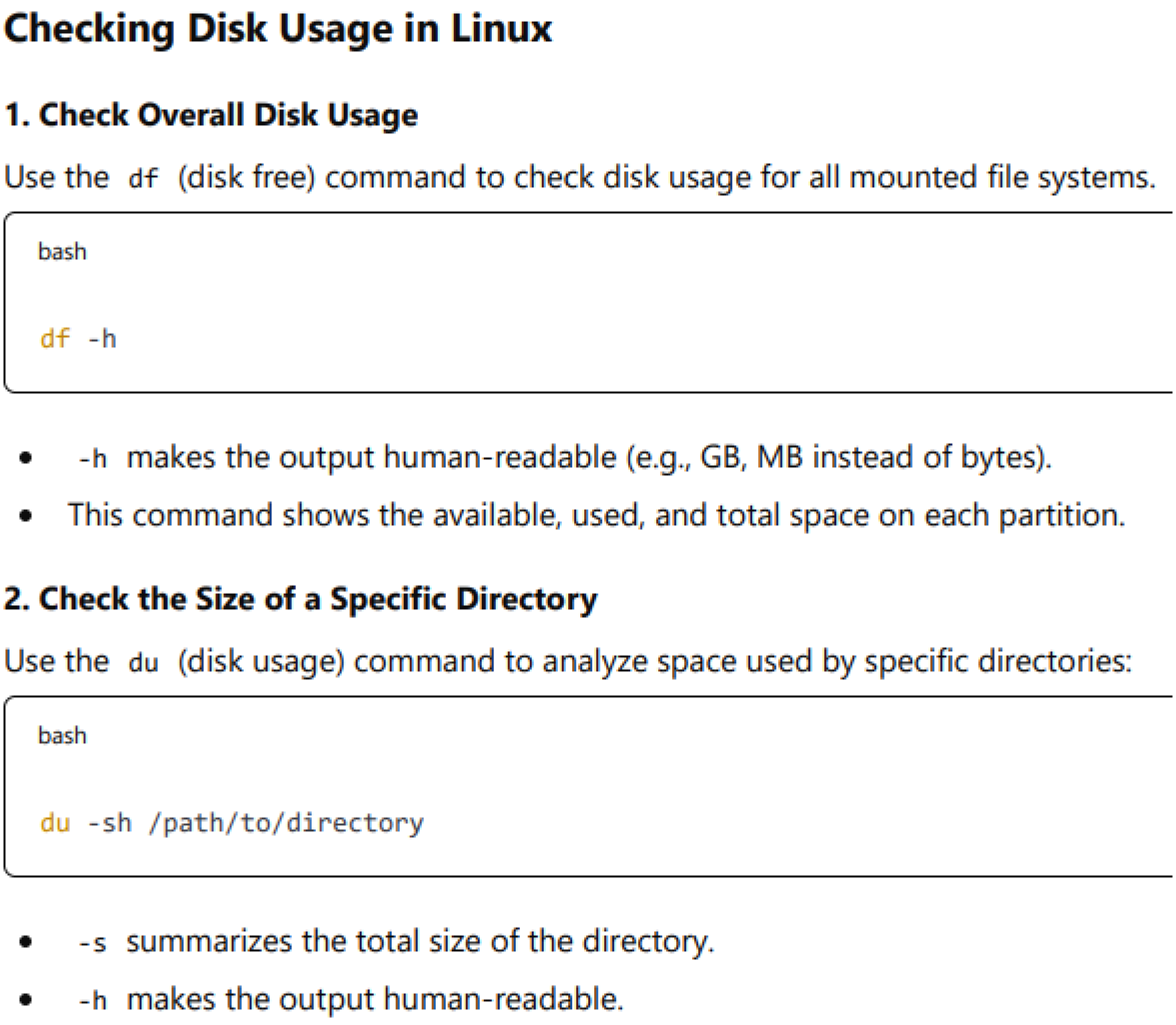
```bash
ping <NFS_Client_IP>
```

Enable Debugging Logs

```bash
sudo journalctl -u nfs-server --no-pager | tail -50
```

Lab 4: Disk Usage Analysis and Cleanup

```
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.
```



Checking Disk Usage in Linux

1. Check Overall Disk Usage

Use the `df` (disk free) command to check disk usage for all mounted file systems.

```bash
df -h
```

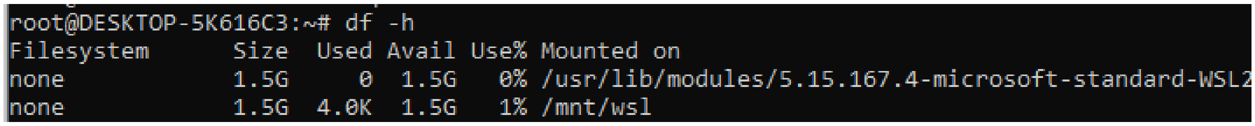
- `-h` makes the output human-readable (e.g., GB, MB instead of bytes).
- This command shows the available, used, and total space on each partition.

2. Check the Size of a Specific Directory

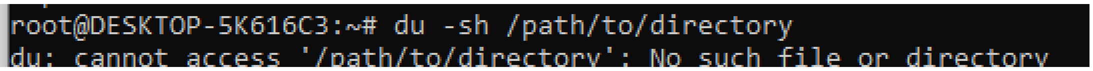
Use the `du` (disk usage) command to analyze space used by specific directories:

```bash
du -sh /path/to/directory
```

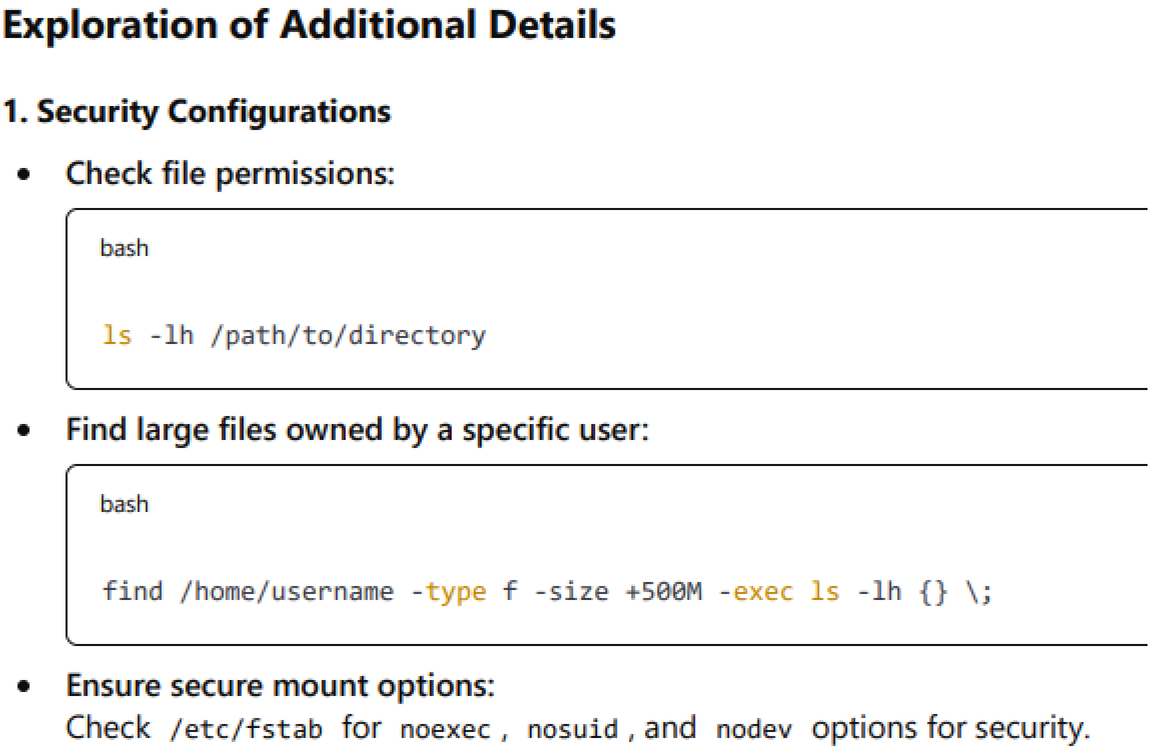
- `-s` summarizes the total size of the directory.
- `-h` makes the output human-readable.



```
root@DESKTOP-5K616C3:~# df -h
Filesystem      Size  Used Avail Use% Mounted on
none            1.5G     0  1.5G   0% /usr/lib/modules/5.15.167.4-microsoft-standard-WSL2
none            1.5G  4.0K  1.5G   1% /mnt/wsl
```



```
root@DESKTOP-5K616C3:~# du -sh /path/to/directory
du: cannot access '/path/to/directory': No such file or directory
```



Exploration of Additional Details

1. Security Configurations

- **Check file permissions:**

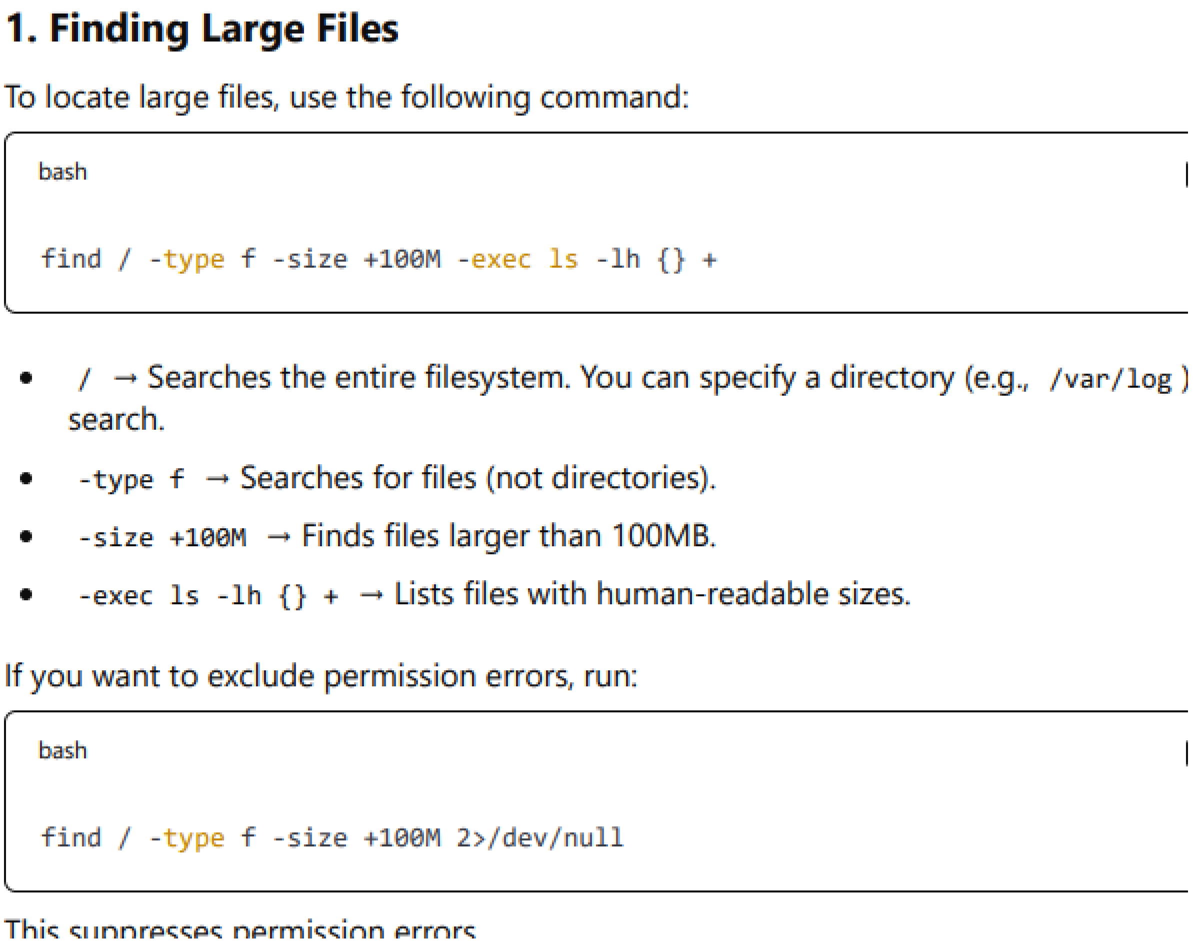
```bash
ls -lh /path/to/directory
```

- **Find large files owned by a specific user:**

```bash
find /home/username -type f -size +500M -exec ls -lh {} \;
```

- **Ensure secure mount options:**
  Check `/etc/fstab` for `noexec`, `nosuid`, and `nodev` options for security.

```
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.
```



1. Finding Large Files

To locate large files, use the following command:

```bash
find / -type f -size +100M -exec ls -lh {} +
```

- `/` → Searches the entire filesystem. You can specify a directory (e.g., `/var/log`) search.
- `-type f` → Searches for files (not directories).
- `-size +100M` → Finds files larger than 100MB.
- `-exec ls -lh {} +` → Lists files with human-readable sizes.

If you want to exclude permission errors, run:

```bash
find / -type f -size +100M 2>/dev/null
```

This suppresses permission errors

If you want to exclude permission errors, run:

```bash
find / -type f -size +100M 2>/dev/null
```

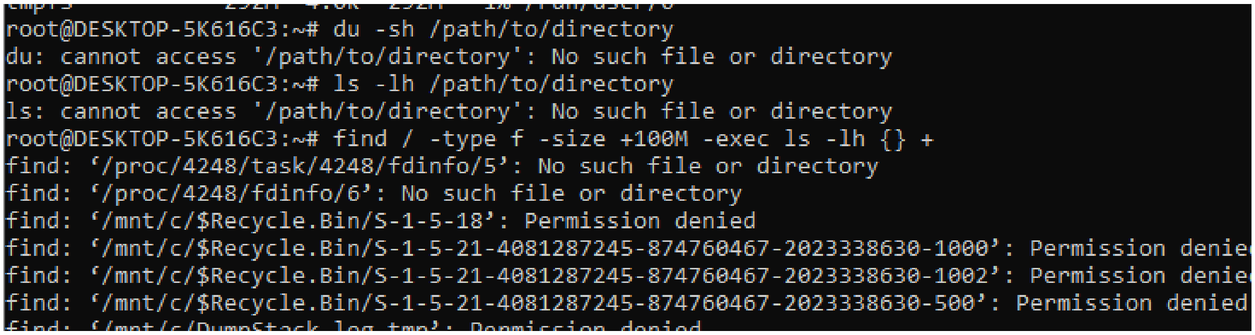
This suppresses permission errors.

Security Considerations

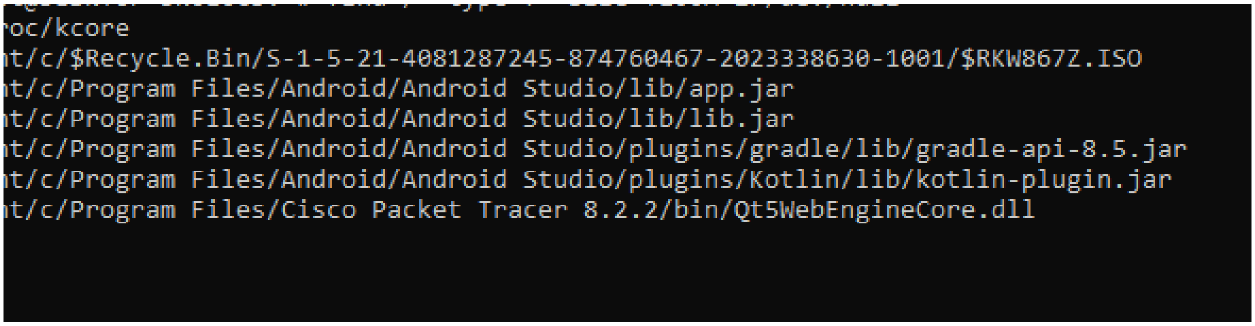
- Running `find /` as a non-root user may result in permission-denied messages. full access:

  ```bash
  sudo find / -type f -size +100M
  ```

- Be cautious with `rm` when deleting large files to avoid unintended deletions.



```
root@DESKTOP-5K616C3:~# du -sh /path/to/directory
du: cannot access '/path/to/directory': No such file or directory
root@DESKTOP-5K616C3:~# ls -lh /path/to/directory
ls: cannot access '/path/to/directory': No such file or directory
root@DESKTOP-5K616C3:~# find / -type f -size +100M -exec ls -lh {} +
find: '/proc/4248/task/4248/fdinfo/5': No such file or directory
find: '/proc/4248/fdinfo/6': No such file or directory
find: '/mnt/c/$Recycle.Bin/S-1-5-18': Permission denied
find: '/mnt/c/$Recycle.Bin/S-1-5-21-4081287245-874760467-2023338630-1000': Permission denie
find: '/mnt/c/$Recycle.Bin/S-1-5-21-4081287245-874760467-2023338630-1002': Permission denie
find: '/mnt/c/$Recycle.Bin/S-1-5-21-4081287245-874760467-2023338630-500': Permission denied
find: '/mnt/c/DumpStack.log.tmp': Permission denied
```



```
oc/kcore
nt/c/$Recycle.Bin/S-1-5-21-4081287245-874760467-2023338630-1001/$RKW867Z.ISO
nt/c/Program Files/Android/Android Studio/lib/app.jar
nt/c/Program Files/Android/Android Studio/lib/lib.jar
nt/c/Program Files/Android/Android Studio/plugins/gradle/lib/gradle-api-8.5.jar
nt/c/Program Files/Android/Android Studio/plugins/Kotlin/lib/kotlin-plugin.jar
nt/c/Program Files/Cisco Packet Tracer 8.2.2/bin/Qt5WebEngineCore.dll
```

2. Using `ncdu` for Disk Usage Analysis

`ncdu` (NCurses Disk Usage) provides an interactive way to analyze disk usage.

Installation

- Debian/Ubuntu:

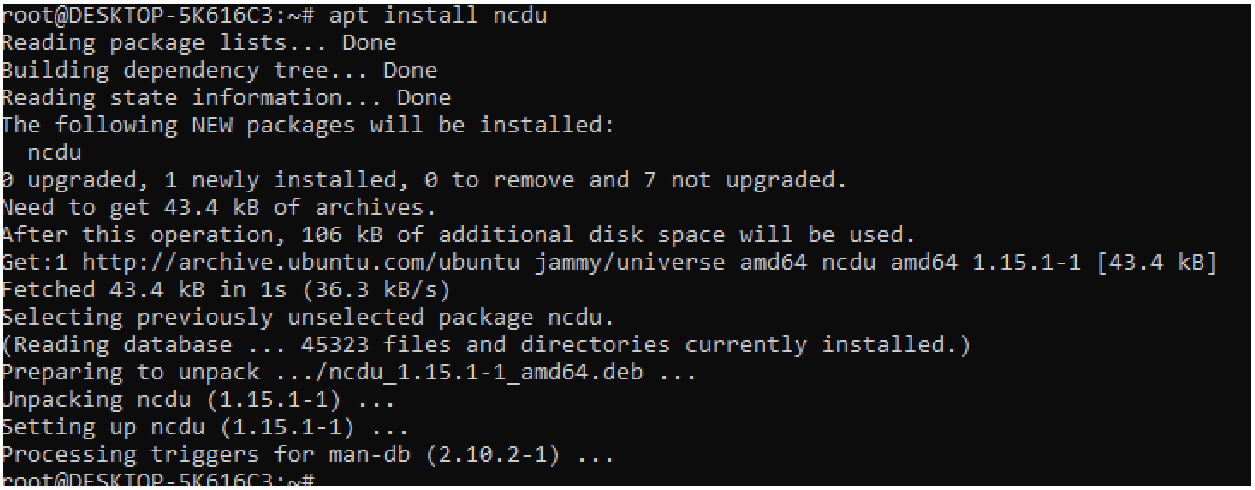
  bash

  ```
  sudo apt install ncdu
  ```

- RHEL/CentOS:

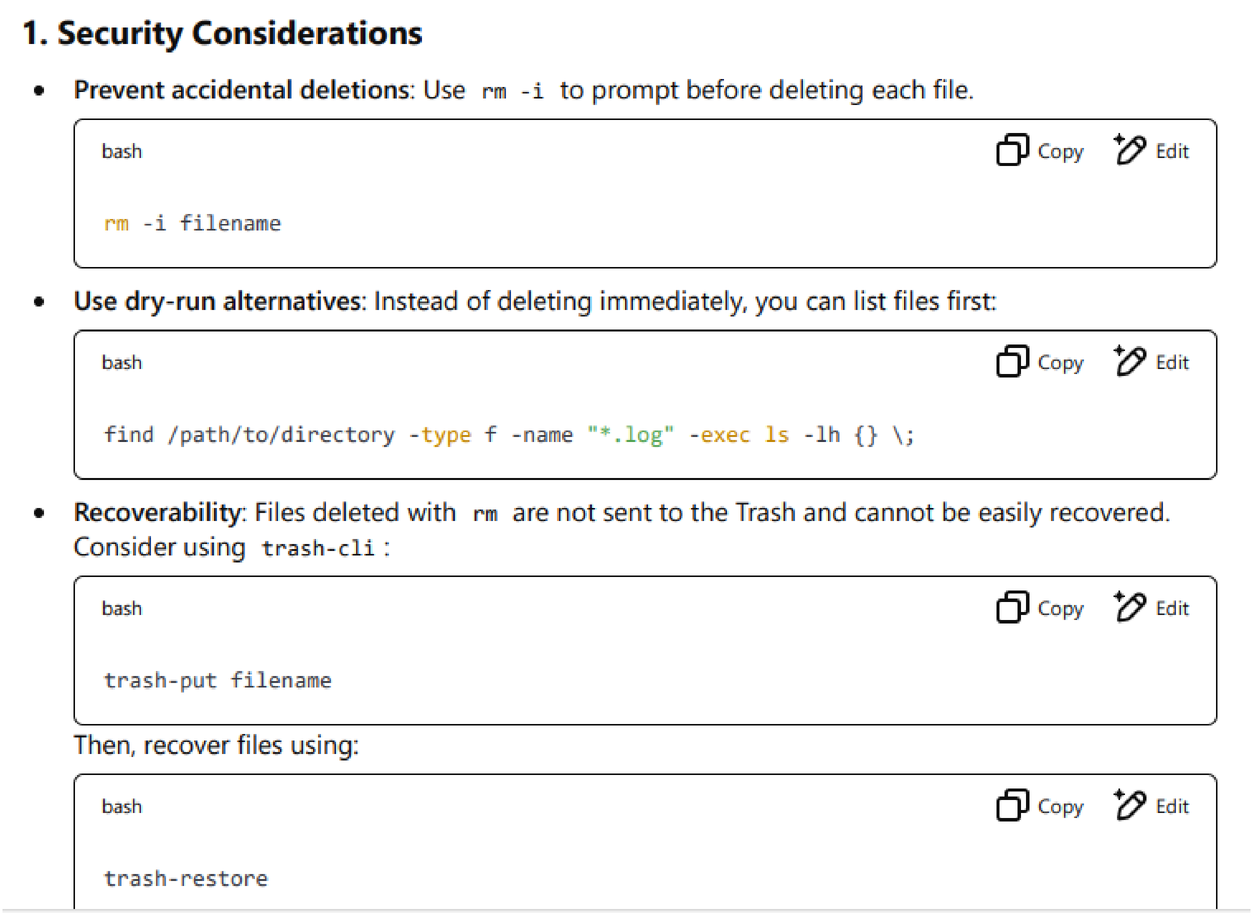
  bash

  ```
  sudo yum install epel-release -y sudo yum install ncdu -y
  ```



```
root@DESKTOP-5K616C3:~# apt install ncdu
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ncdu
0 upgraded, 1 newly installed, 0 to remove and 7 not upgraded.
Need to get 43.4 kB of archives.
After this operation, 106 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ncdu amd64 1.15.1-1 [43.4 kB]
Fetched 43.4 kB in 1s (36.3 kB/s)
Selecting previously unselected package ncdu.
(Reading database ... 45323 files and directories currently installed.)
Preparing to unpack .../ncdu_1.15.1-1_amd64.deb ...
Unpacking ncdu (1.15.1-1) ...
Setting up ncdu (1.15.1-1) ...
Processing triggers for man-db (2.10.2-1) ...
root@DESKTOP-5K616C3:~#
```

```
3.      Clean Up Old Files:

  #     Identify and delete unnecessary files using the rm command.

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



1. Security Considerations

- **Prevent accidental deletions**: Use `rm -i` to prompt before deleting each file.

```bash
rm -i filename
```

- **Use dry-run alternatives**: Instead of deleting immediately, you can list files first:

```bash
find /path/to/directory -type f -name "*.log" -exec ls -lh {} \;
```

- **Recoverability**: Files deleted with `rm` are not sent to the Trash and cannot be easily recovered. Consider using `trash-cli`:

```bash
trash-put filename
```

Then, recover files using:

```bash
trash-restore
```

2. Performance Tuning

- **Find and remove large files**

```bash
find /path/to/directory -type f -size +100M -exec rm -f {} \;
```

- **Delete files older than a certain number of days**

```bash
find /path/to/directory -type f -mtime +30 -exec rm -f {} \;
```

- **Clear system logs**

```bash
sudo journalctl --vacuum-time=7d # Keep logs for 7 days sudo journalctl --vacuum-size=500M # Limit logs to 500MB
```

- **Permission Denied**: Run with `sudo` if necessary:

```bash
sudo rm -rf /path/to/file
```

- **"Argument list too long" error**: Use `find` instead of `rm *`:

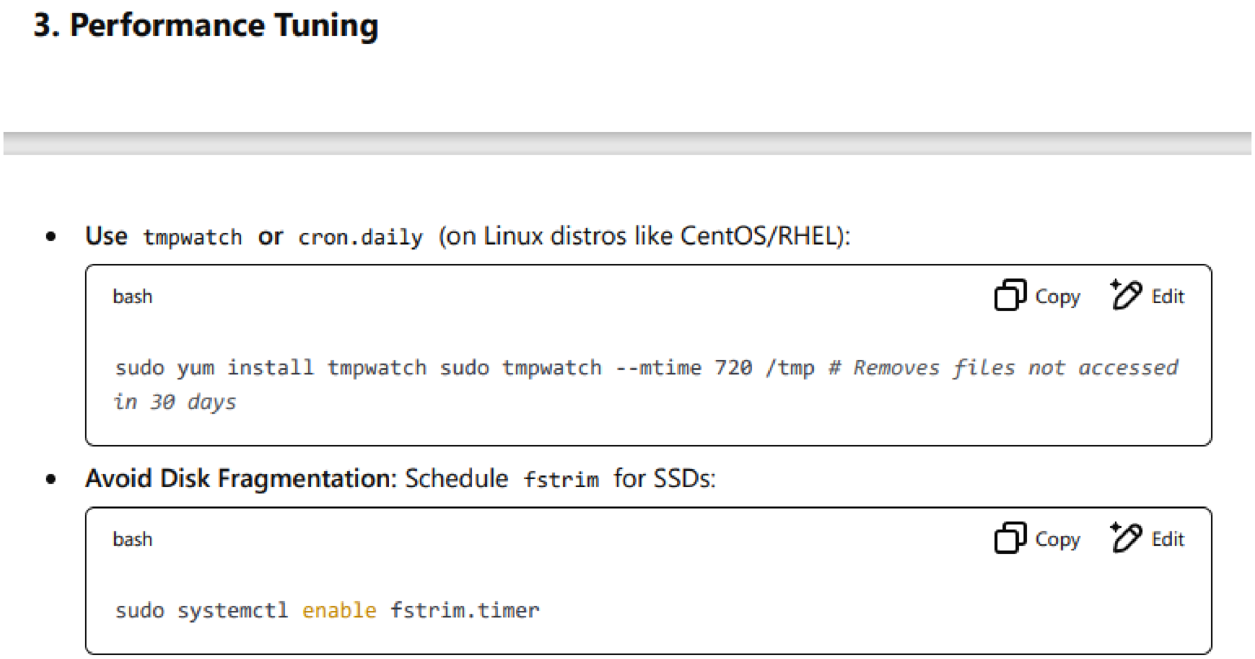
```bash
find /path/to/directory -type f -delete
```

- **Files reappearing after deletion**: Some system processes may be recreating files. Check running processes:

```bash
lsof | grep "/path/to/file"
```

4. Automate Cleanup:

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



3. Performance Tuning

- **Use** `tmpwatch` **or** `cron.daily` (on Linux distros like CentOS/RHEL):

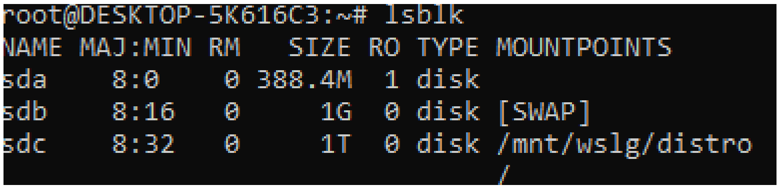
```bash
sudo yum install tmpwatch sudo tmpwatch --mtime 720 /tmp # Removes files not accessed in 30 days
```

- **Avoid Disk Fragmentation:** Schedule `fstrim` for SSDs:

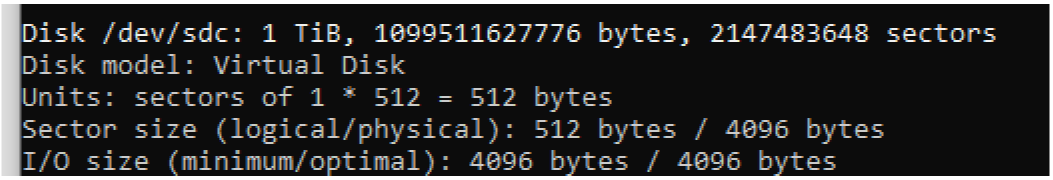
```bash
sudo systemctl enable fstrim.timer
```

Lab 5: LVM (Logical Volume Management) Setup

```
   1) Create Physical Volume (PV):

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



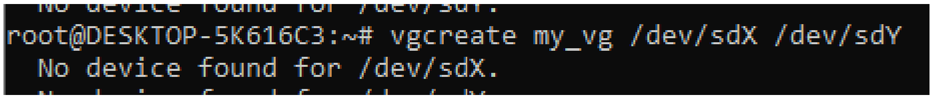
```
root@DESKTOP-5K616C3:~# lsblk
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 388.4M  1 disk
sdb    8:16   0     1G  0 disk [SWAP]
sdc    8:32   0     1T  0 disk /mnt/wslg/distro
                                /
```



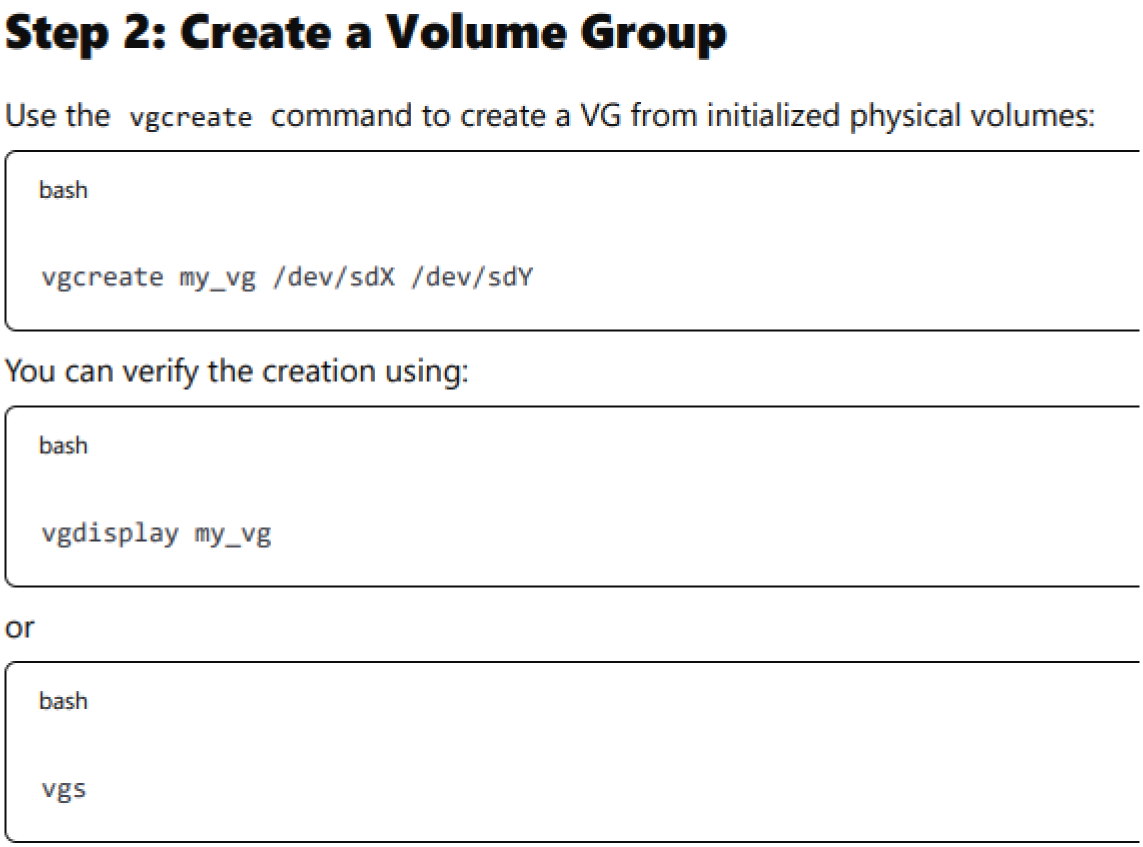
```
Disk /dev/sdc: 1 TiB, 1099511627776 bytes, 2147483648 sectors
Disk model: Virtual Disk
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 4096 bytes
I/O size (minimum/optimal): 4096 bytes / 4096 bytes
```

```
2.      Create Volume Group (VG):

  #     Use vgcreate to create a volume group (e.g., vg_data).
```

Create a Volume Group



```
  No device found for /dev/sdY.
root@DESKTOP-5K616C3:~# vgcreate my_vg /dev/sdX /dev/sdY
  No device found for /dev/sdX.
  No device found for /dev/sdY.
```



Step 2: Create a Volume Group

Use the `vgcreate` command to create a VG from initialized physical volumes:

```bash
vgcreate my_vg /dev/sdX /dev/sdY
```

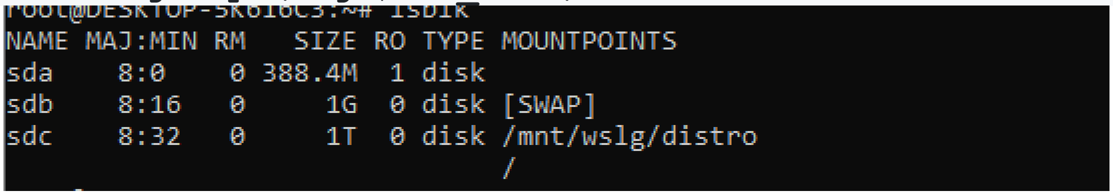
You can verify the creation using:

```bash
vgdisplay my_vg
```

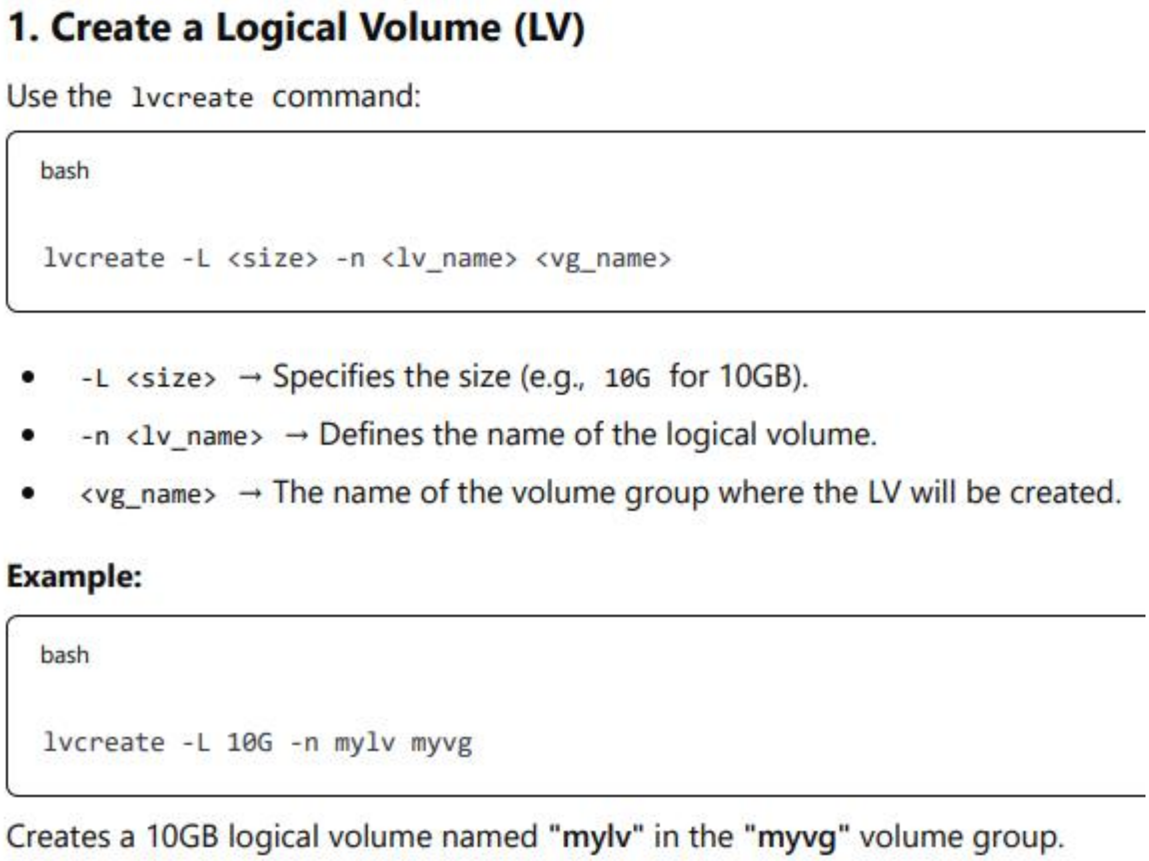
or

```bash
vgs
```

```
3.    Create Logical Volume (LV):

  #   Use lvcreate to create a logical volume from the
volume group (e.g., lv_data).
```



```
root@DESKTOP-5K616C3:~# lsblk
NAME MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
sda    8:0    0 388.4M  1 disk
sdb    8:16   0     1G  0 disk [SWAP]
sdc    8:32   0     1T  0 disk /mnt/wslg/distro
                               /
```



1. Create a Logical Volume (LV)

Use the `lvcreate` command:

```bash
lvcreate -L <size> -n <lv_name> <vg_name>
```

- `-L <size>` → Specifies the size (e.g., `10G` for 10GB).
- `-n <lv_name>` → Defines the name of the logical volume.
- `<vg_name>` → The name of the volume group where the LV will be created.

Example:

```bash
lvcreate -L 10G -n mylv myvg
```

Creates a 10GB logical volume named **"mylv"** in the **"myvg"** volume group.

```
4.    Create File System:

  #   Format the logical volume with a file system
(e.g., mkfs.ext4 /dev/vg_data/lv_data).
```

2. Verify the Logical Volume

Check if the logical volume was created successfully:

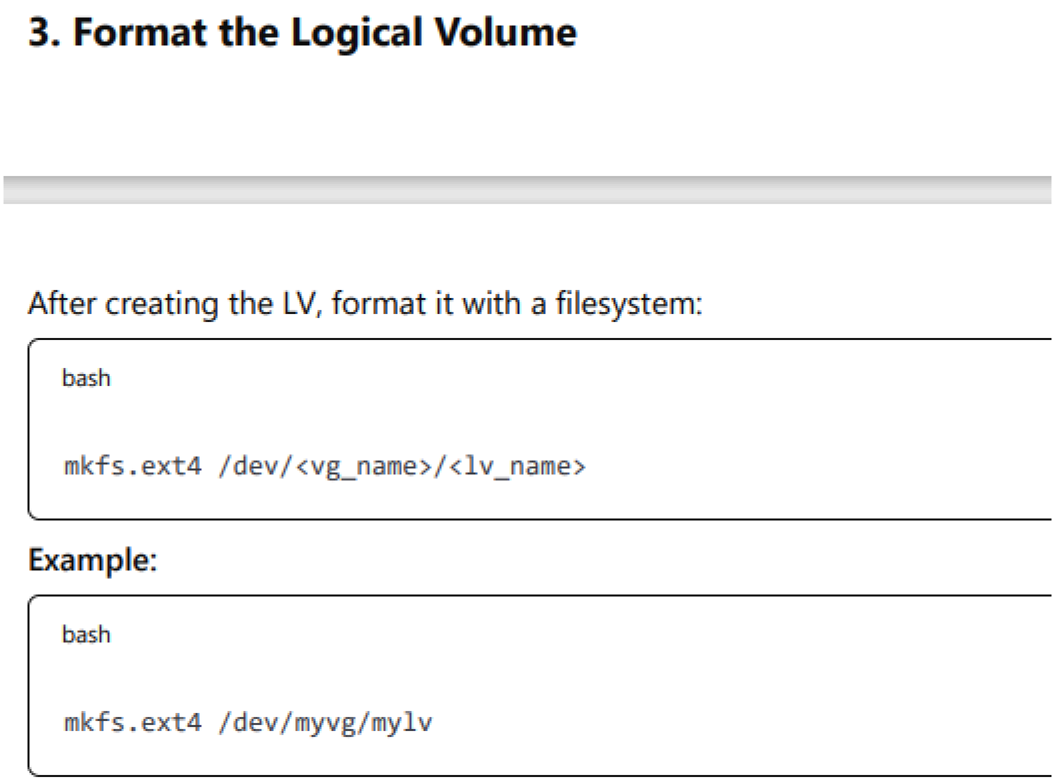
bash

```
lvdisplay
```

or

bash

```
lsblk
```



3. Format the Logical Volume

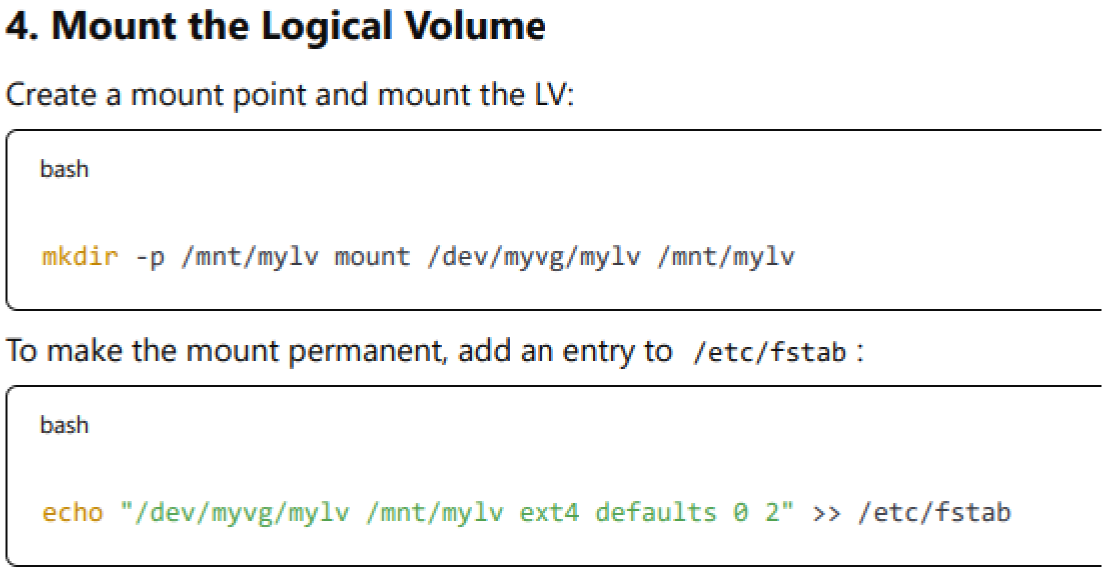
After creating the LV, format it with a filesystem:

```bash
mkfs.ext4 /dev/<vg_name>/<lv_name>
```

Example:

```bash
mkfs.ext4 /dev/myvg/mylv
```

```
5.    Mount and Extend Logical Volume:

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



4. Mount the Logical Volume

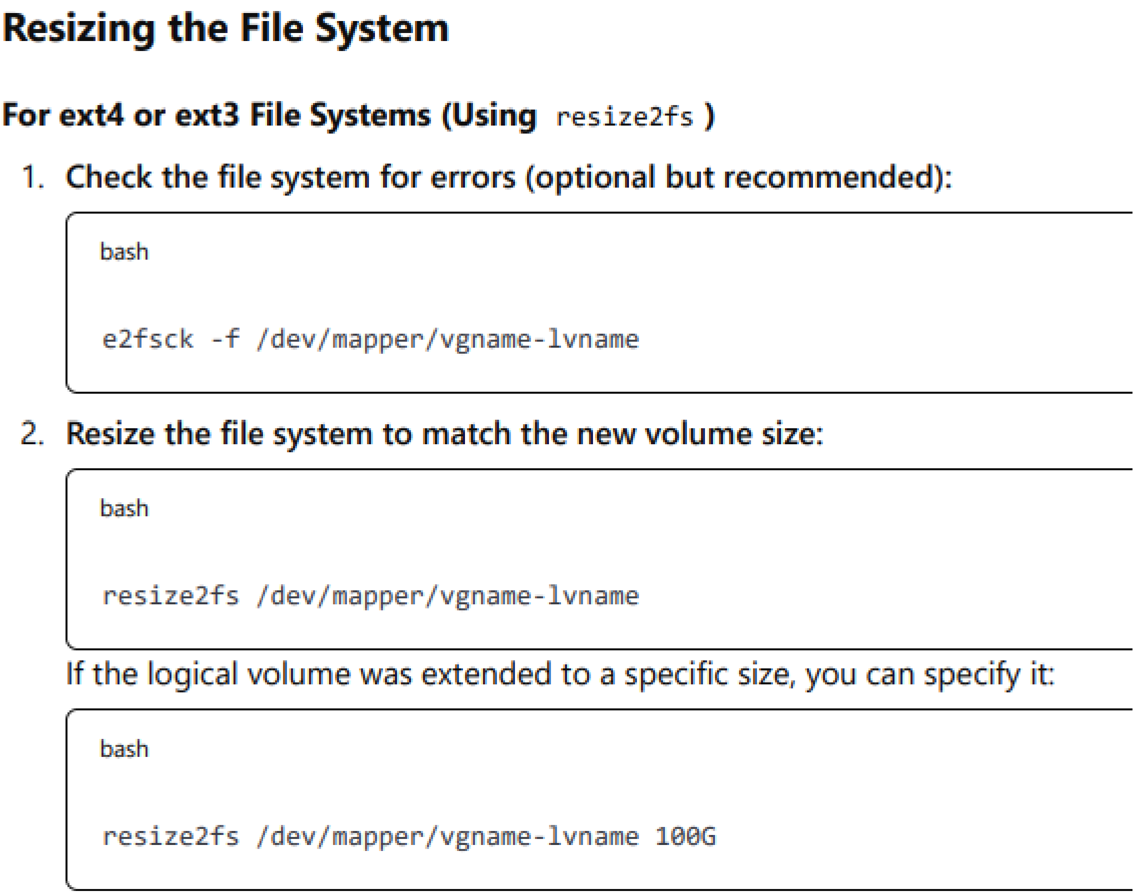
Create a mount point and mount the LV:

```bash
mkdir -p /mnt/mylv mount /dev/myvg/mylv /mnt/mylv
```

To make the mount permanent, add an entry to `/etc/fstab` :

```bash
echo "/dev/myvg/mylv /mnt/mylv ext4 defaults 0 2" >> /etc/fstab
```

```
6.      Resize File System:

  #     Use resize2fs or xfs_growfs to resize the file system after extending
the logical volume.
```



Resizing the File System

For ext4 or ext3 File Systems (Using `resize2fs`)

1. Check the file system for errors (optional but recommended):

   ```bash
   e2fsck -f /dev/mapper/vgname-lvname
   ```

2. Resize the file system to match the new volume size:

   ```bash
   resize2fs /dev/mapper/vgname-lvname
   ```

   If the logical volume was extended to a specific size, you can specify it:

   ```bash
   resize2fs /dev/mapper/vgname-lvname 100G
   ```

For XFS File Systems (Using `xfs_growfs`)

1. Simply run the following command (XFS does not need a file system check):

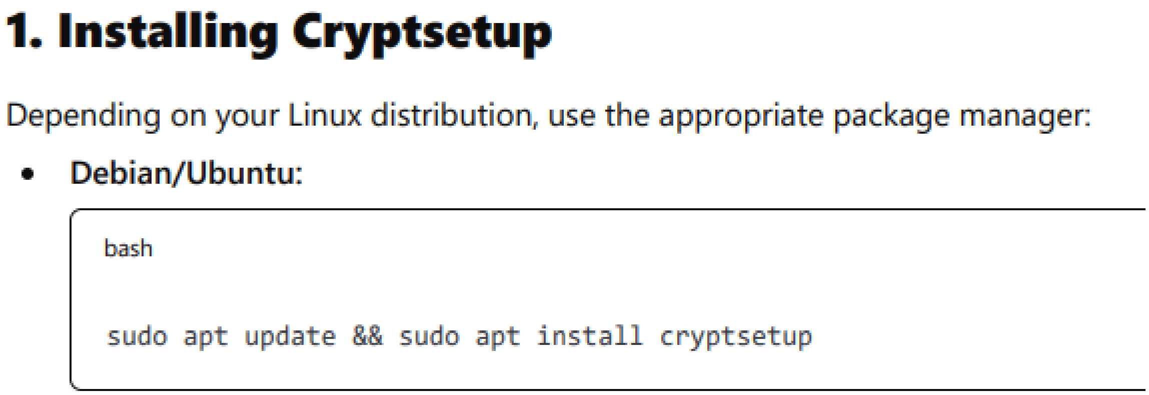
   ```bash
   xfs_growfs /mount/point
   ```

   Example:

   ```bash
   xfs_growfs /data
   ```

Lab 6: Disk Encryption with LUKS

```
1.      Install Cryptsetup:

  #     Install cryptsetup to manage LUKS encryption
```

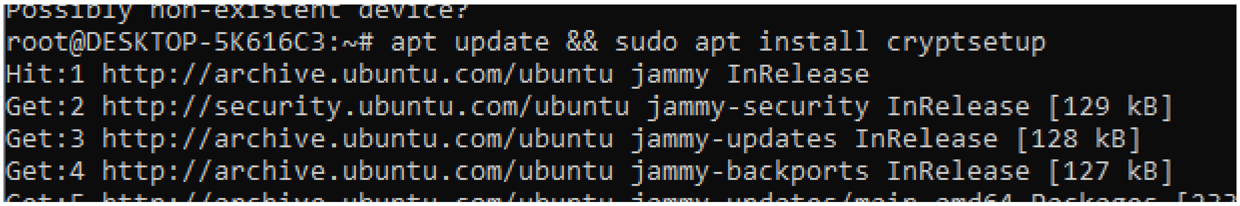


1. Installing Cryptsetup

Depending on your Linux distribution, use the appropriate package manager:

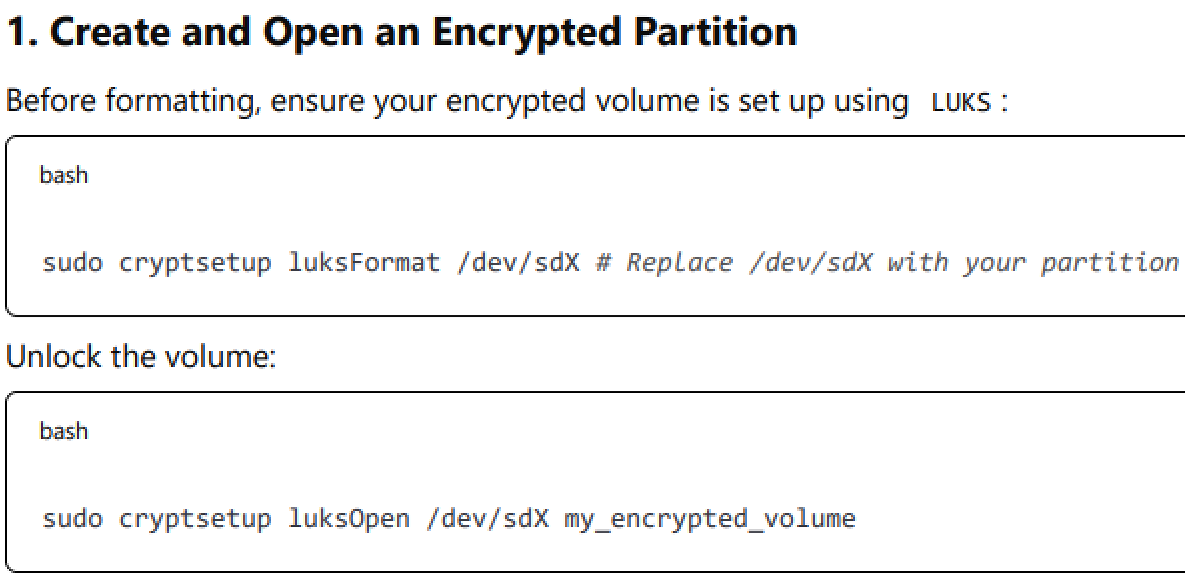
- **Debian/Ubuntu:**

```bash
sudo apt update && sudo apt install cryptsetup
```



```
Possibly non-existent device?
root@DESKTOP-5K616C3:~# apt update && sudo apt install cryptsetup
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [222
```

```
2.    Create an Encrypted Partition:

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



1. Create and Open an Encrypted Partition

Before formatting, ensure your encrypted volume is set up using `LUKS`:

bash

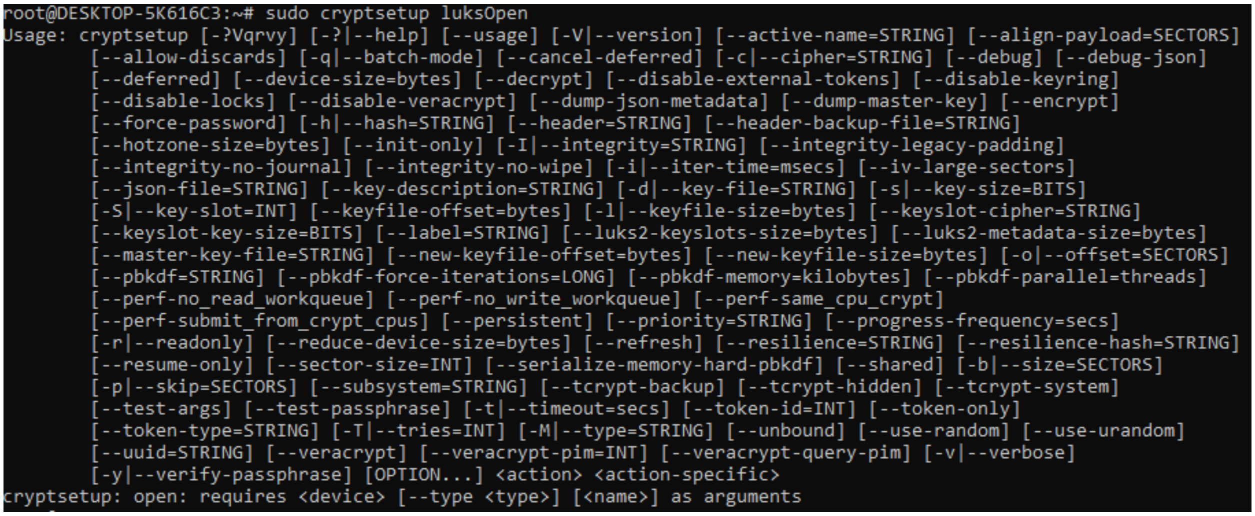
```
sudo cryptsetup luksFormat /dev/sdX # Replace /dev/sdX with your partition
```

Unlock the volume:

bash

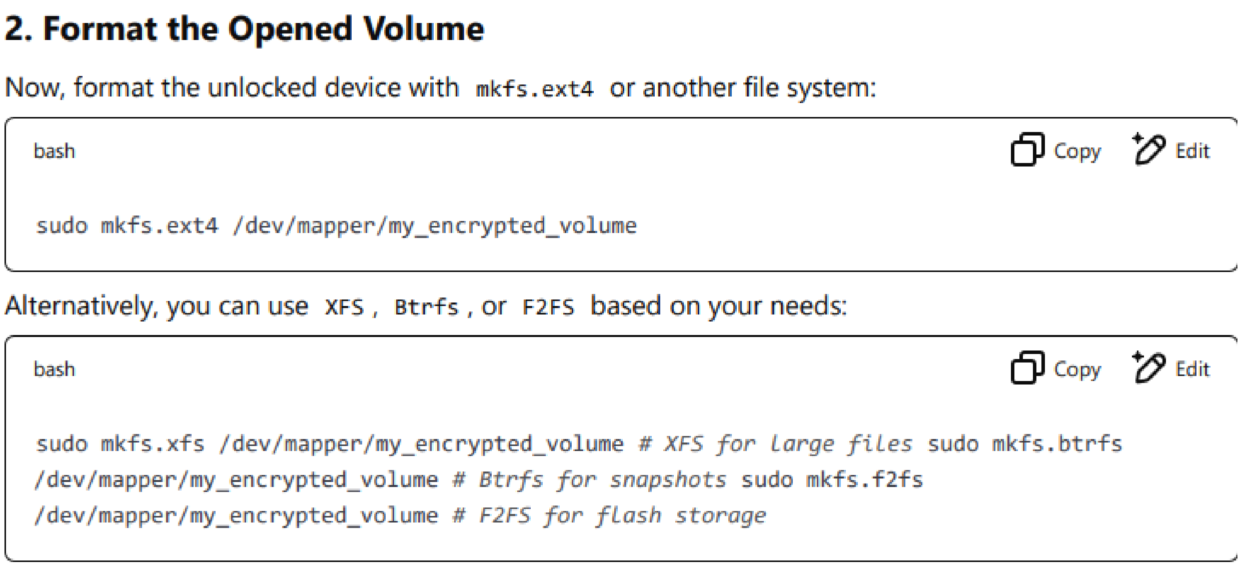
```
sudo cryptsetup luksOpen /dev/sdX my_encrypted_volume
```

```
3.Open Encrypted Volume:

  #  Use cryptsetup luksOpen /dev/sdb1
encrypted_data to open the encrypted volume.
```



```
root@DESKTOP-5K616C3:~# sudo cryptsetup luksOpen
Usage: cryptsetup [-?Vqrvy] [-?|--help] [--usage] [-V|--version] [--active-name=STRING] [--align-payload=SECTORS]
        [--allow-discards] [-q|--batch-mode] [--cancel-deferred] [-c|--cipher=STRING] [--debug] [--debug-json]
        [--deferred] [--device-size=bytes] [--decrypt] [--disable-external-tokens] [--disable-keyring]
        [--disable-locks] [--disable-veracrypt] [--dump-json-metadata] [--dump-master-key] [--encrypt]
        [--force-password] [-h|--hash=STRING] [--header=STRING] [--header-backup-file=STRING]
        [--hotzone-size=bytes] [--init-only] [-I|--integrity=STRING] [--integrity-legacy-padding]
        [--integrity-no-journal] [--integrity-no-wipe] [-i|--iter-time=msecs] [--iv-large-sectors]
        [--json-file=STRING] [--key-description=STRING] [-d|--key-file=STRING] [-s|--key-size=BITS]
        [-S|--key-slot=INT] [--keyfile-offset=bytes] [-l|--keyfile-size=bytes] [--keyslot-cipher=STRING]
        [--keyslot-key-size=BITS] [--label=STRING] [--luks2-keyslots-size=bytes] [--luks2-metadata-size=bytes]
        [--master-key-file=STRING] [--new-keyfile-offset=bytes] [--new-keyfile-size=bytes] [-o|--offset=SECTORS]
        [--pbkdf=STRING] [--pbkdf-force-iterations=LONG] [--pbkdf-memory=kilobytes] [--pbkdf-parallel=threads]
        [--perf-no_read_workqueue] [--perf-no_write_workqueue] [--perf-same_cpu_crypt]
        [--perf-submit_from_crypt_cpus] [--persistent] [--priority=STRING] [--progress-frequency=secs]
        [-r|--readonly] [--reduce-device-size=bytes] [--refresh] [--resilience=STRING] [--resilience-hash=STRING]
        [--resume-only] [--sector-size=INT] [--serialize-memory-hard-pbkdf] [--shared] [-b|--size=SECTORS]
        [-p|--skip=SECTORS] [--subsystem=STRING] [--tcrypt-backup] [--tcrypt-hidden] [--tcrypt-system]
        [--test-args] [--test-passphrase] [-t|--timeout=secs] [--token-id=INT] [--token-only]
        [--token-type=STRING] [-T|--tries=INT] [-M|--type=STRING] [--unbound] [--use-random] [--use-urandom]
        [--uuid=STRING] [--veracrypt] [--veracrypt-pim=INT] [--veracrypt-query-pim] [-v|--verbose]
        [-y|--verify-passphrase] [OPTION...] <action> <action-specific>
cryptsetup: open: requires <device> [--type <type>] [<name>] as arguments
```

```
4.      Create File System on Encrypted Partition:

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



2. Format the Opened Volume

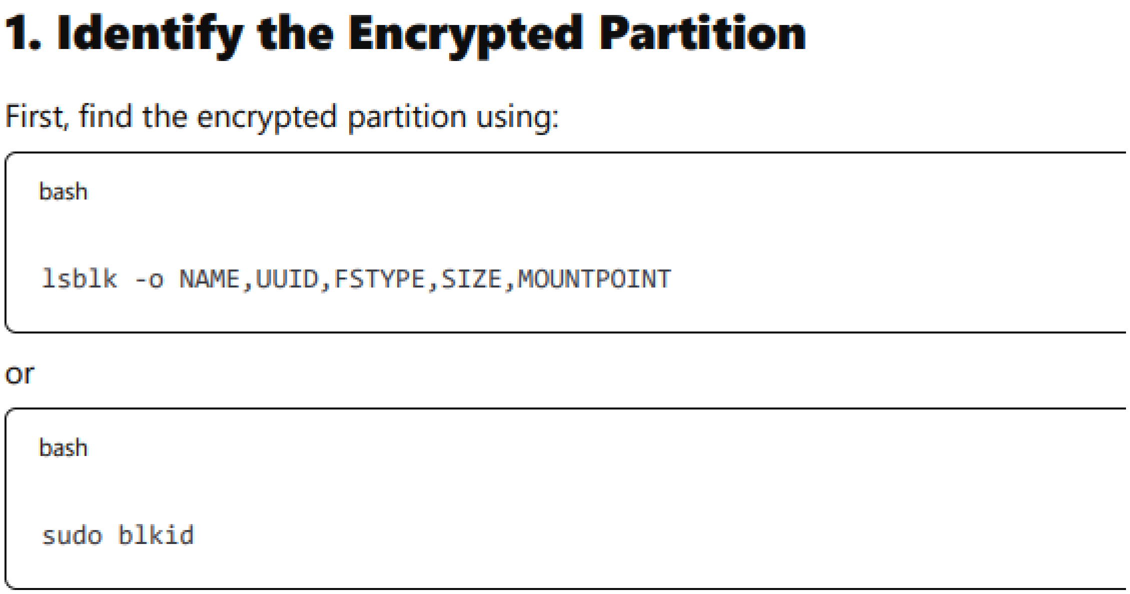
Now, format the unlocked device with `mkfs.ext4` or another file system:

```bash
sudo mkfs.ext4 /dev/mapper/my_encrypted_volume
```

Alternatively, you can use `XFS`, `Btrfs`, or `F2FS` based on your needs:

```bash
sudo mkfs.xfs /dev/mapper/my_encrypted_volume # XFS for large files sudo mkfs.btrfs
/dev/mapper/my_encrypted_volume # Btrfs for snapshots sudo mkfs.f2fs
/dev/mapper/my_encrypted_volume # F2FS for flash storage
```

```
5.Mount and Configure Auto-Mount:

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



1. Identify the Encrypted Partition

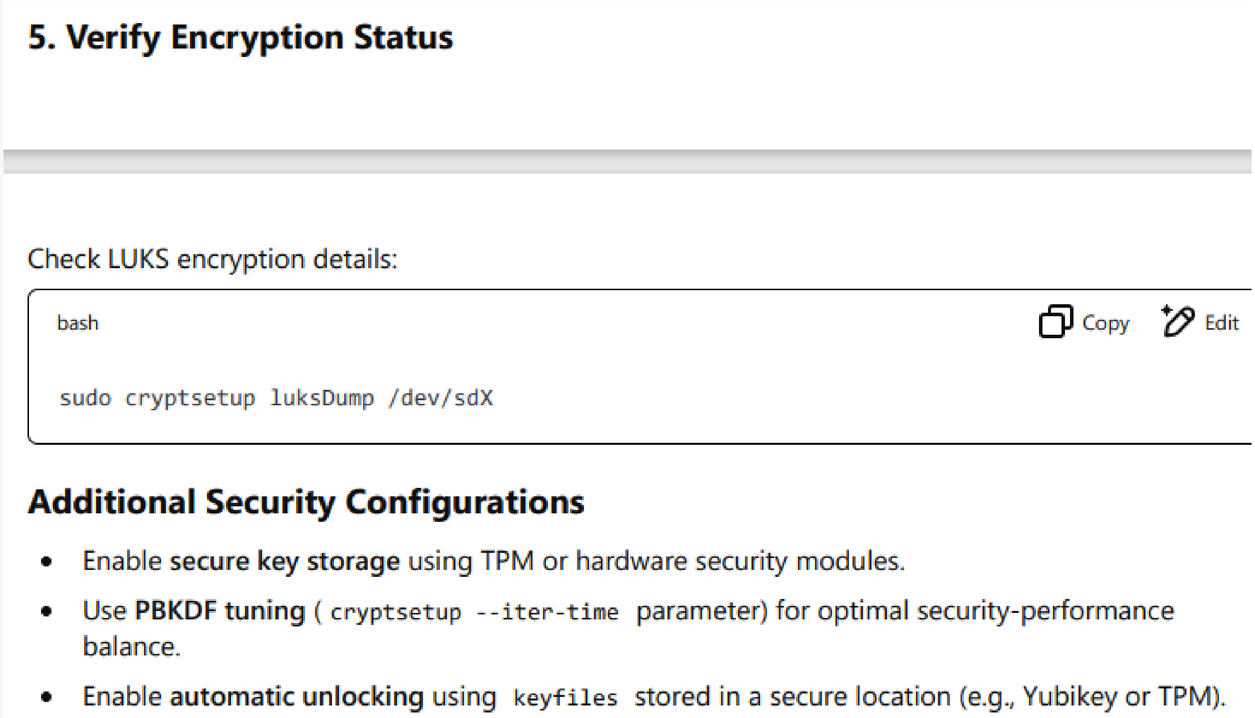
First, find the encrypted partition using:

```bash
lsblk -o NAME,UUID,FSTYPE,SIZE,MOUNTPOINT
```

or

```bash
sudo blkid
```

```
6.   Verify Encryption:

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



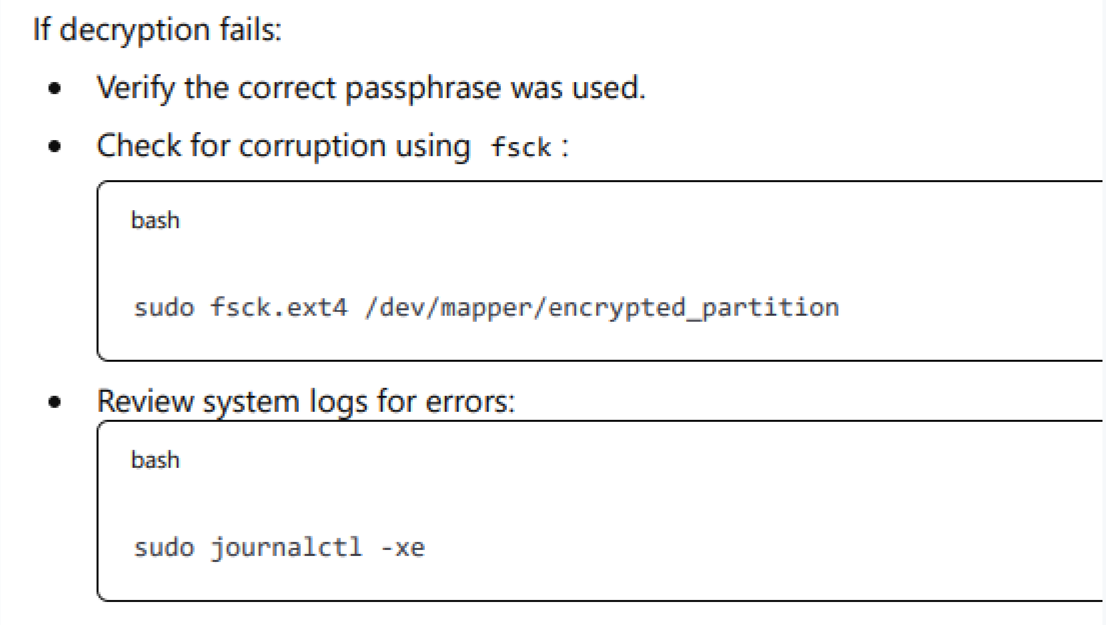
5. Verify Encryption Status

Check LUKS encryption details:

```bash
sudo cryptsetup luksDump /dev/sdX
```

Additional Security Configurations

- Enable **secure key storage** using TPM or hardware security modules.
- Use **PBKDF tuning** (`cryptsetup --iter-time` parameter) for optimal security-performance balance.
- Enable **automatic unlocking** using `keyfiles` stored in a secure location (e.g., Yubikey or TPM).



If decryption fails:

- Verify the correct passphrase was used.
- Check for corruption using `fsck` :

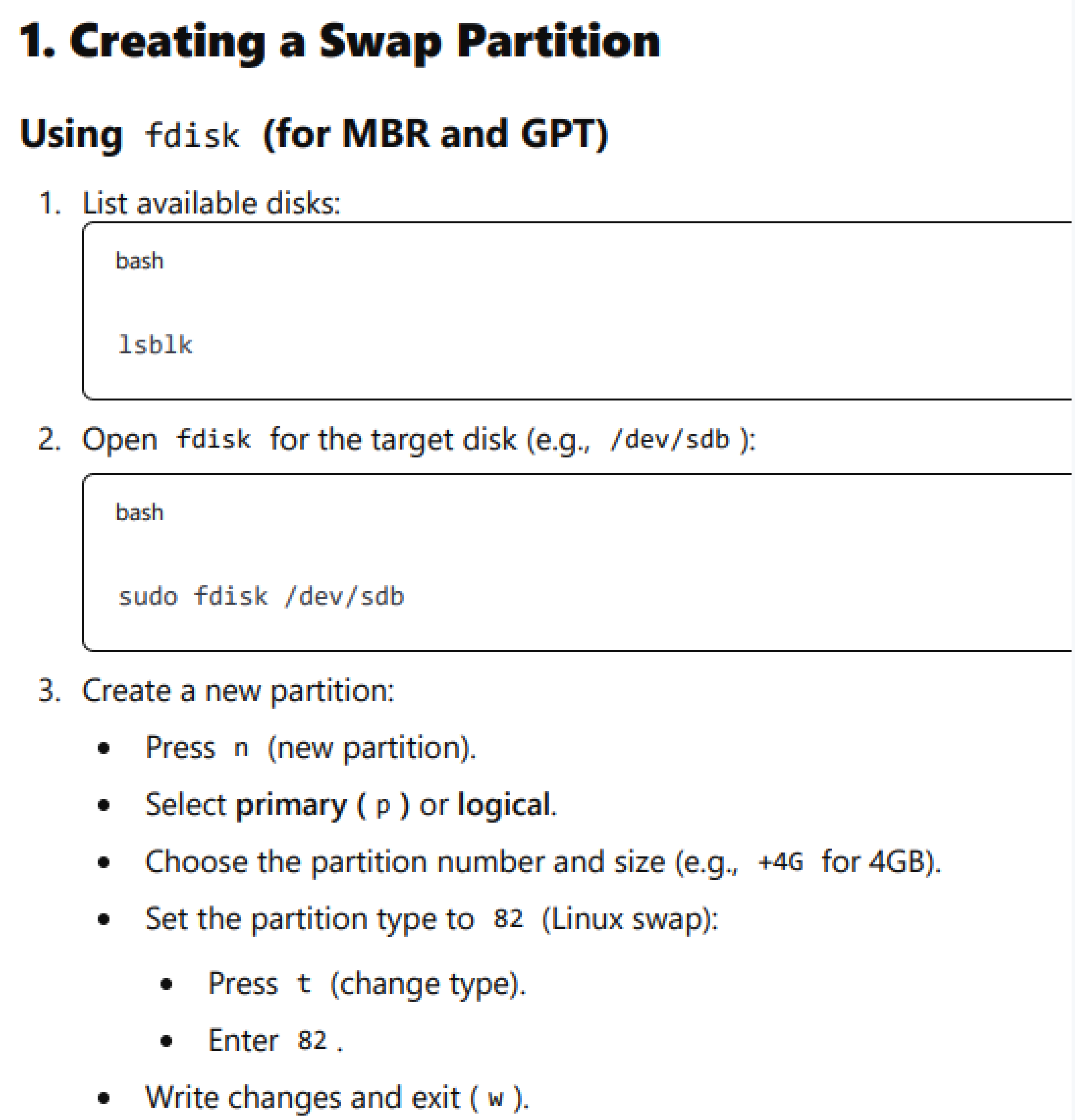
  ```bash
  sudo fsck.ext4 /dev/mapper/encrypted_partition
  ```

- Review system logs for errors:

  ```bash
  sudo journalctl -xe
  ```

Lab 7: Creating and Managing Swap Space

```
1.      Create a Swap Partition:

  #     Use fdisk or parted to create a swap partition.

  #     Format the partition with mkswap.
```



1. Creating a Swap Partition

Using `fdisk` (for MBR and GPT)

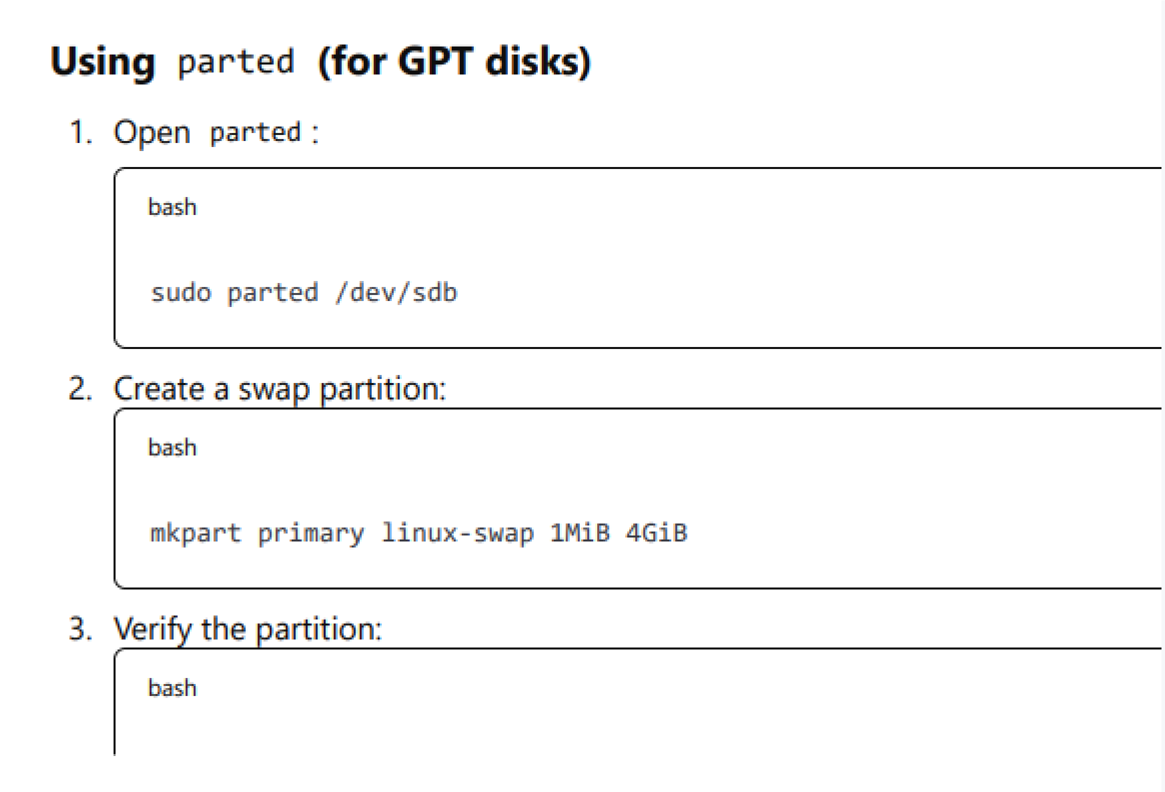
1. List available disks:

```bash
lsblk
```

2. Open `fdisk` for the target disk (e.g., `/dev/sdb`):

```bash
sudo fdisk /dev/sdb
```

3. Create a new partition:
   - Press `n` (new partition).
   - Select **primary** (`p`) or **logical**.
   - Choose the partition number and size (e.g., `+4G` for 4GB).
   - Set the partition type to `82` (Linux swap):
     - Press `t` (change type).
     - Enter `82` .
   - Write changes and exit (`w`).



Using `parted` (for GPT disks)

1. Open `parted`:

   bash

   ```bash
   sudo parted /dev/sdb
   ```

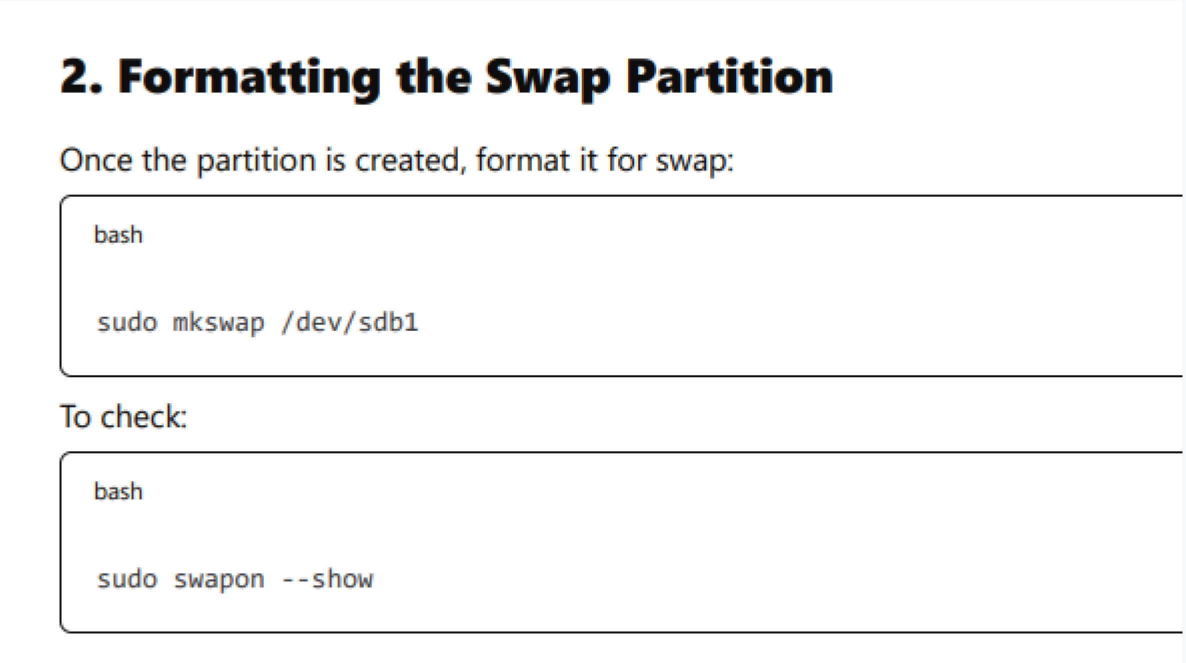
2. Create a swap partition:

   bash

   ```bash
   mkpart primary linux-swap 1MiB 4GiB
   ```

3. Verify the partition:

   bash



2. Formatting the Swap Partition

Once the partition is created, format it for swap:

bash

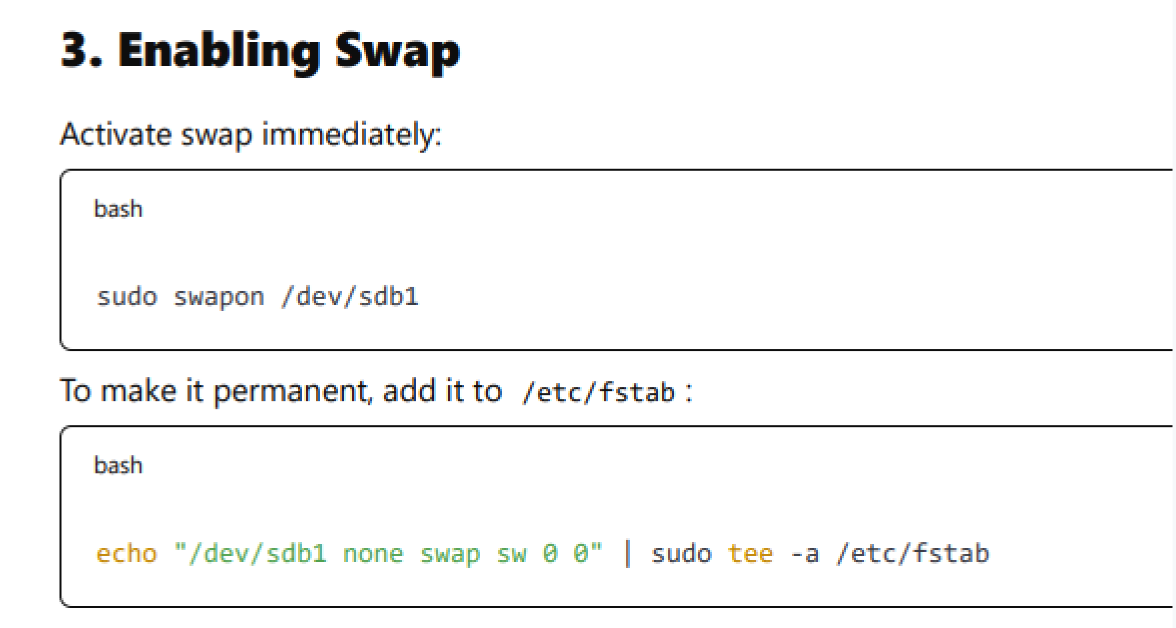
```bash
sudo mkswap /dev/sdb1
```

To check:

bash

```bash
sudo swapon --show
```

```
2.    Enable Swap:

  #   Enable the swap space using swapon /dev/sdb1.
```



3. Enabling Swap

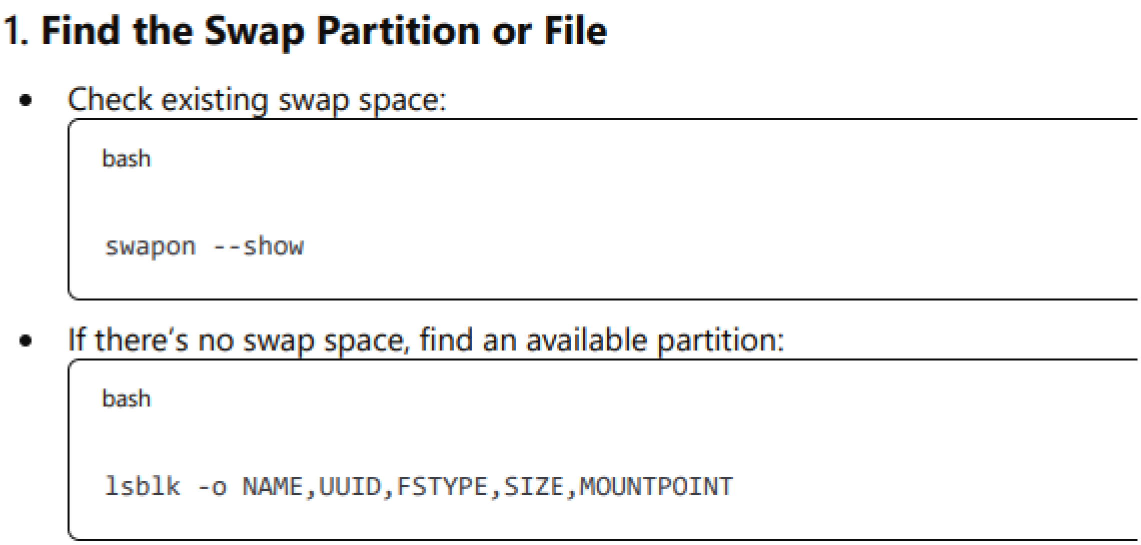
Activate swap immediately:

```bash
sudo swapon /dev/sdb1
```

To make it permanent, add it to `/etc/fstab` :

```bash
echo "/dev/sdb1 none swap sw 0 0" | sudo tee -a /etc/fstab
```

```
3.    Add Swap to /etc/fstab:

  #   Edit /etc/fstab to ensure that the swap partition is
mounted automatically at boot.
```



1. Find the Swap Partition or File

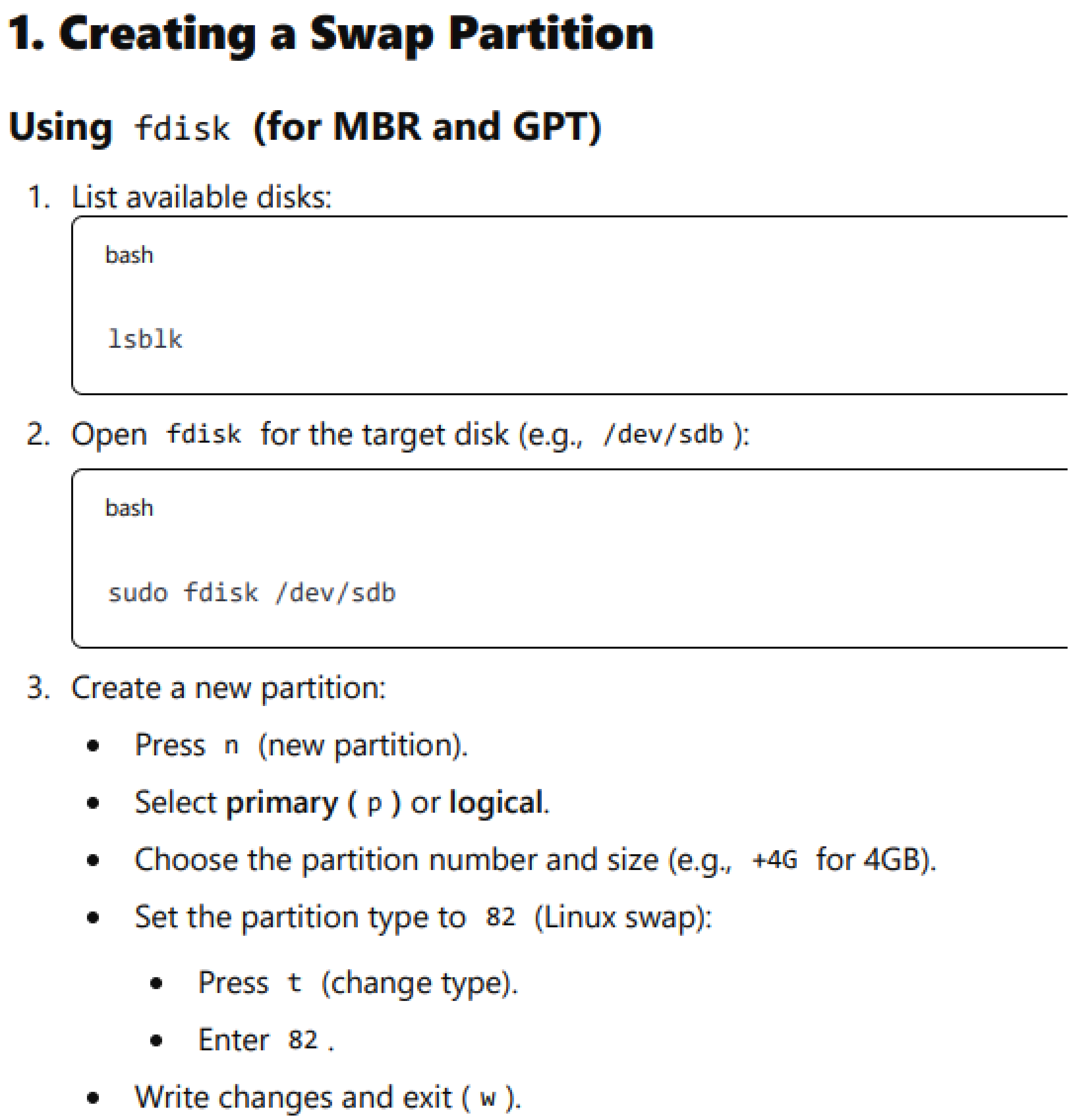
- Check existing swap space:

```bash
swapon --show
```

- If there's no swap space, find an available partition:

```bash
lsblk -o NAME,UUID,FSTYPE,SIZE,MOUNTPOINT
```

```
4.    Create Swap File:

  #  Create a swap file using dd if=/dev/zero
of=/swapfile bs=1M count=1024 and enable it using
swapon /swapfile.
```



1. Creating a Swap Partition

Using `fdisk` (for MBR and GPT)

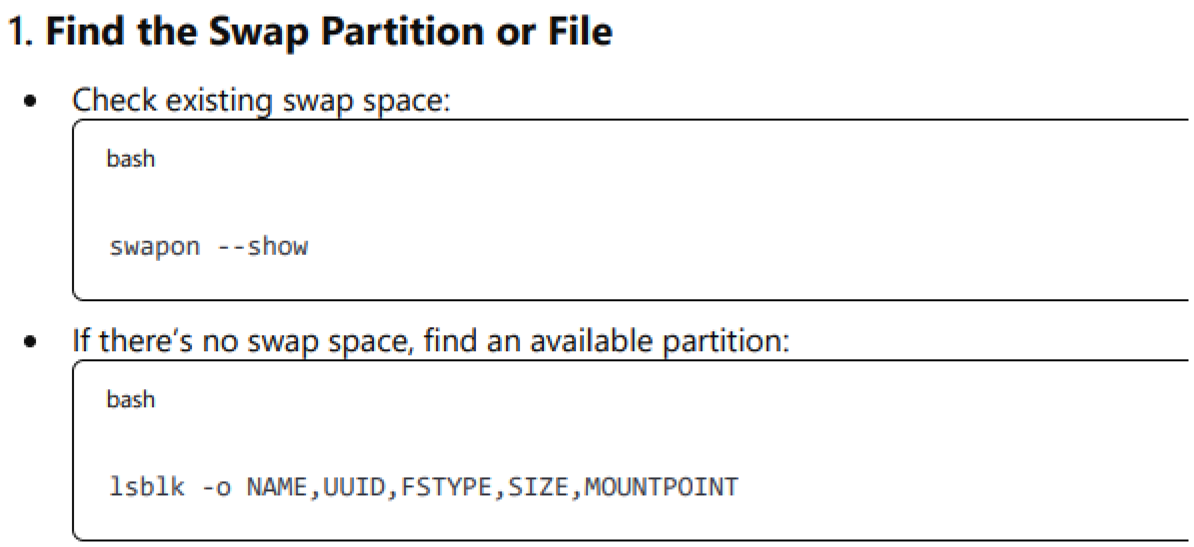
1. List available disks:

```bash
lsblk
```

2. Open `fdisk` for the target disk (e.g., `/dev/sdb`):

```bash
sudo fdisk /dev/sdb
```

3. Create a new partition:
   - Press `n` (new partition).
   - Select **primary (** `p` **)** or **logical**.
   - Choose the partition number and size (e.g., `+4G` for 4GB).
   - Set the partition type to `82` (Linux swap):
     - Press `t` (change type).
     - Enter `82`.
   - Write changes and exit (`w`).



1. Find the Swap Partition or File

- Check existing swap space:

```bash
swapon --show
```

- If there's no swap space, find an available partition:

```bash
lsblk -o NAME,UUID,FSTYPE,SIZE,MOUNTPOINT
```

5. Verify Swap:

# Use swapon -s to verify the active swap spaces.

# Check system memory and swap usage using free -h.

1. Verify Active Swap Spaces

Use the following command to list active swap partitions or files:

bash

```
swapon -s
```

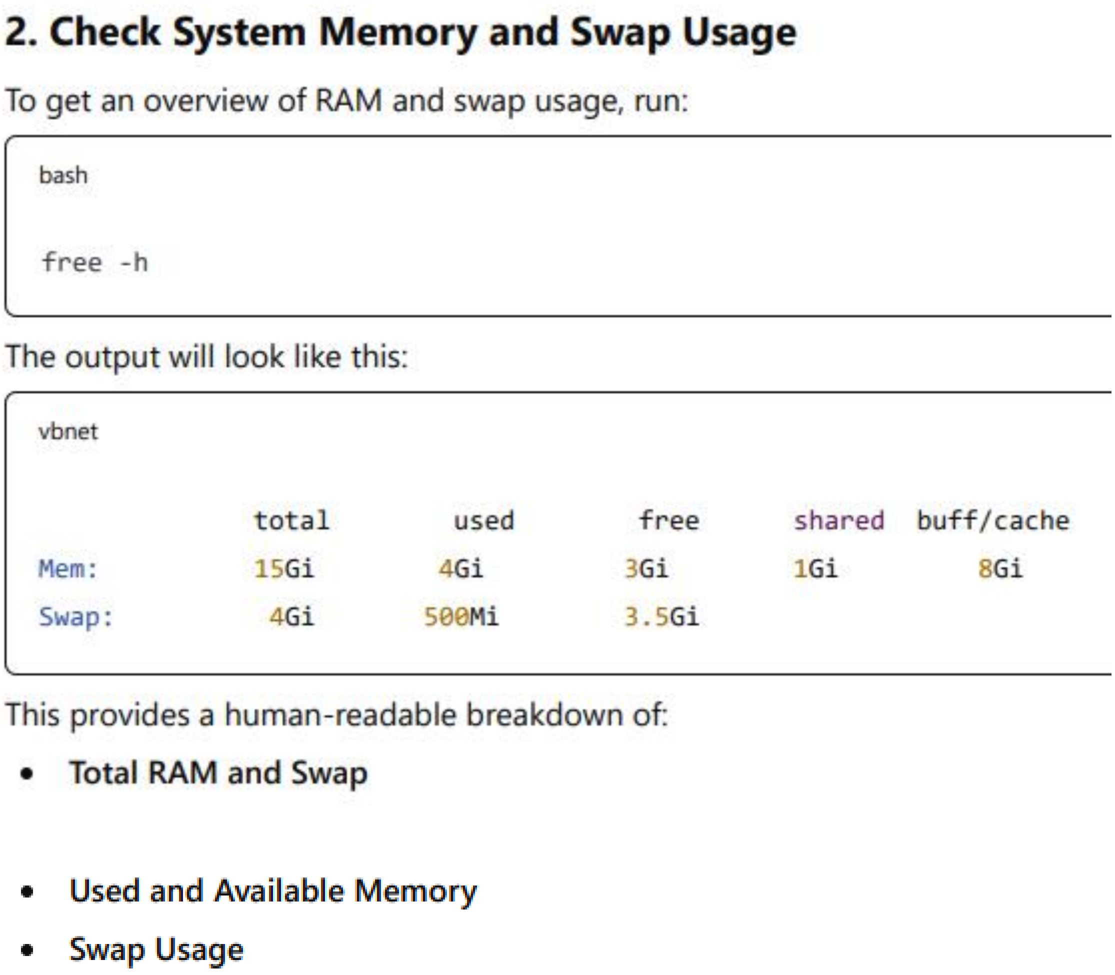
or for a more detailed output:

bash

```
swapon --show
```

This will display details such as:

- **Filename**: Path of the swap file or partition.
- **Type**: Whether it is a partition or a file.
- **Size**: Total swap size.
- **Used**: Amount of swap in use.
- **Priority**: The priority level for swap usage.



2. Check System Memory and Swap Usage

To get an overview of RAM and swap usage, run:

bash

```
free -h
```

The output will look like this:

vbnet

```
              total        used        free      shared  buff/cache
Mem:           15Gi         4Gi         3Gi         1Gi         8Gi
Swap:           4Gi       500Mi       3.5Gi
```

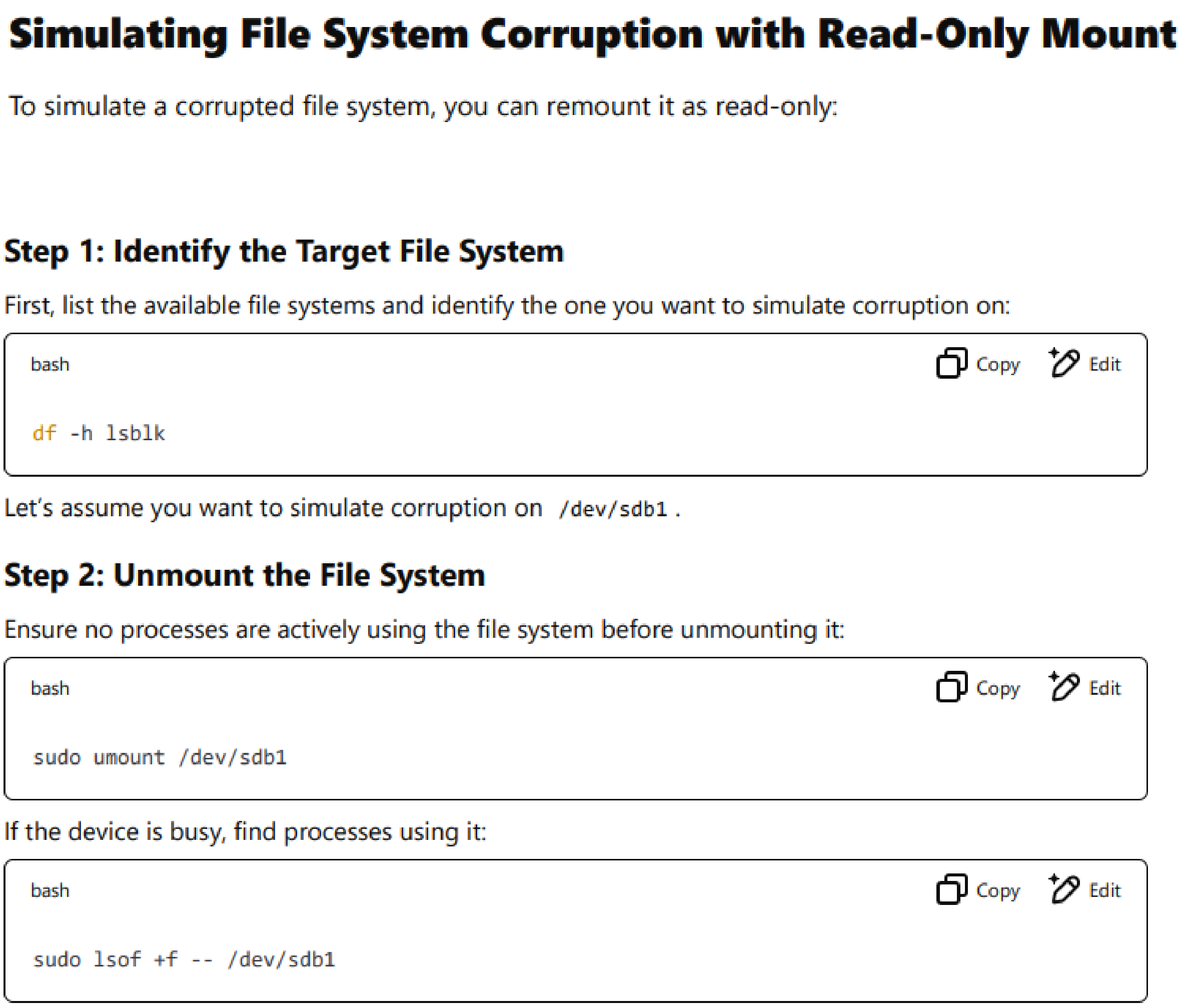
This provides a human-readable breakdown of:

- **Total RAM and Swap**

- **Used and Available Memory**
- **Swap Usage**

Lab 8: Filesystem 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.
```



Simulating File System Corruption with Read-Only Mount

To simulate a corrupted file system, you can remount it as read-only:

Step 1: Identify the Target File System

First, list the available file systems and identify the one you want to simulate corruption on:

```bash
df -h lsblk
```

Let's assume you want to simulate corruption on `/dev/sdb1` .

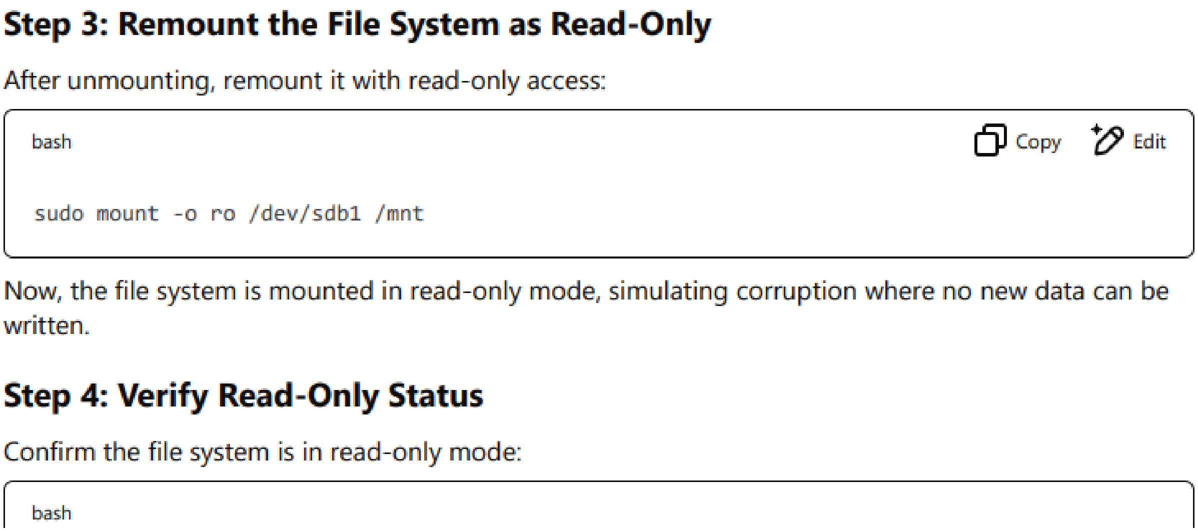
Step 2: Unmount the File System

Ensure no processes are actively using the file system before unmounting it:

```bash
sudo umount /dev/sdb1
```

If the device is busy, find processes using it:

```bash
sudo lsof +f -- /dev/sdb1
```



Step 3: Remount the File System as Read-Only

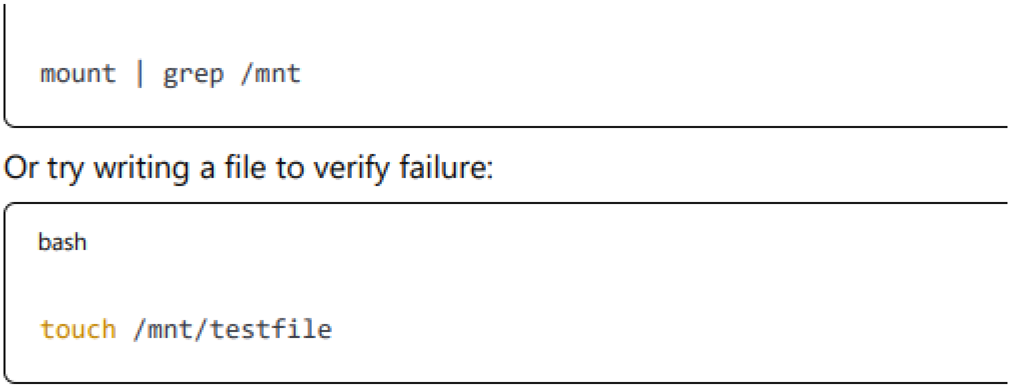
After unmounting, remount it with read-only access:

```bash
sudo mount -o ro /dev/sdb1 /mnt
```

Now, the file system is mounted in read-only mode, simulating corruption where no new data can be written.

Step 4: Verify Read-Only Status

Confirm the file system is in read-only mode:

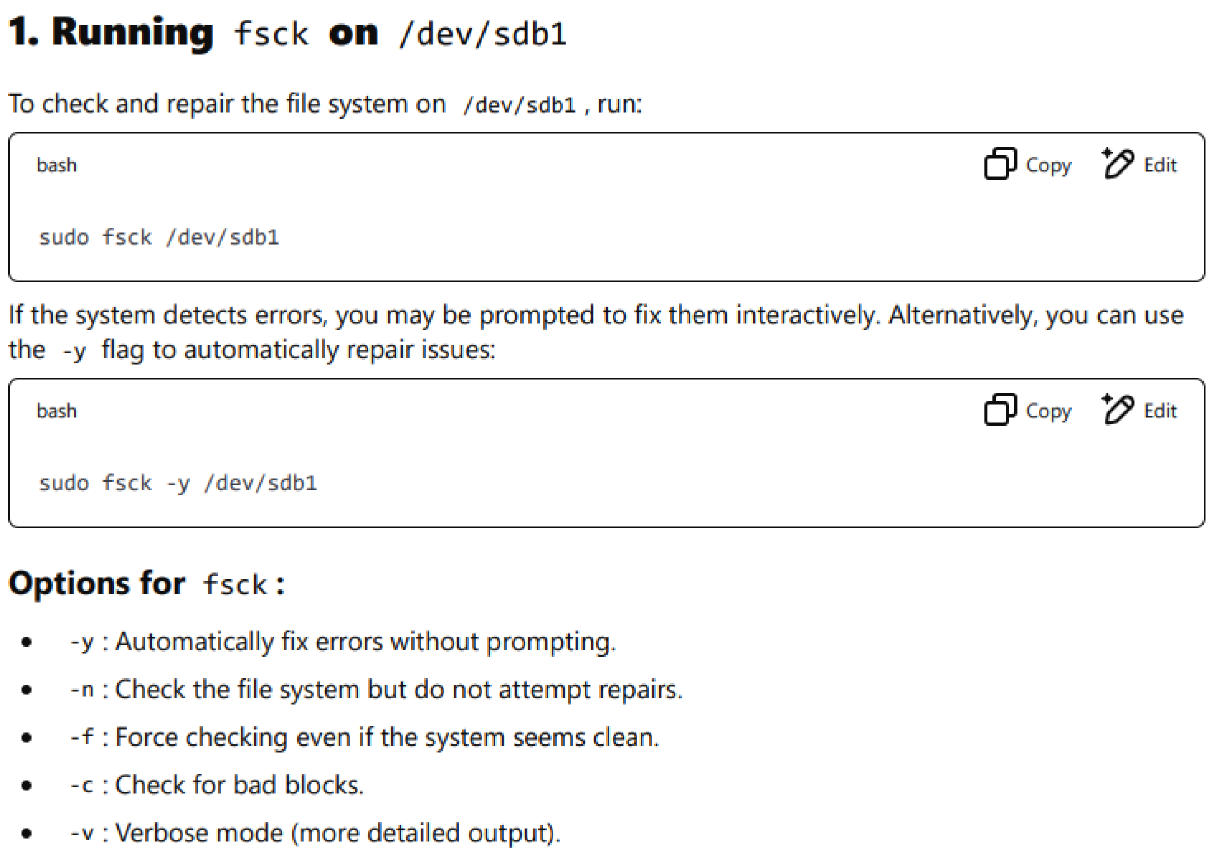


```bash
mount | grep /mnt
```

Or try writing a file to verify failure:

```bash
touch /mnt/testfile
```

```
2.      Run fsck:

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



1. Running `fsck` on `/dev/sdb1`

To check and repair the file system on `/dev/sdb1`, run:

```bash
sudo fsck /dev/sdb1
```

If the system detects errors, you may be prompted to fix them interactively. Alternatively, you can use the `-y` flag to automatically repair issues:

```bash
sudo fsck -y /dev/sdb1
```

Options for `fsck`:

- `-y` : Automatically fix errors without prompting.
- `-n` : Check the file system but do not attempt repairs.
- `-f` : Force checking even if the system seems clean.
- `-c` : Check for bad blocks.
- `-v` : Verbose mode (more detailed output).

```
3.Repair Options:

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

Common `fsck` Options for Repair:

1. `fsck -A` **(Check All File Systems)**
   - Checks all file systems listed in `/etc/fstab`.
   - Typically used during system boot to verify multiple disks.
   - Example:

     ```bash
     fsck -A
     ```

   - You can combine it with `-p` (preen mode) for automatic fixing:

     ```bash
     fsck -A -p
     ```

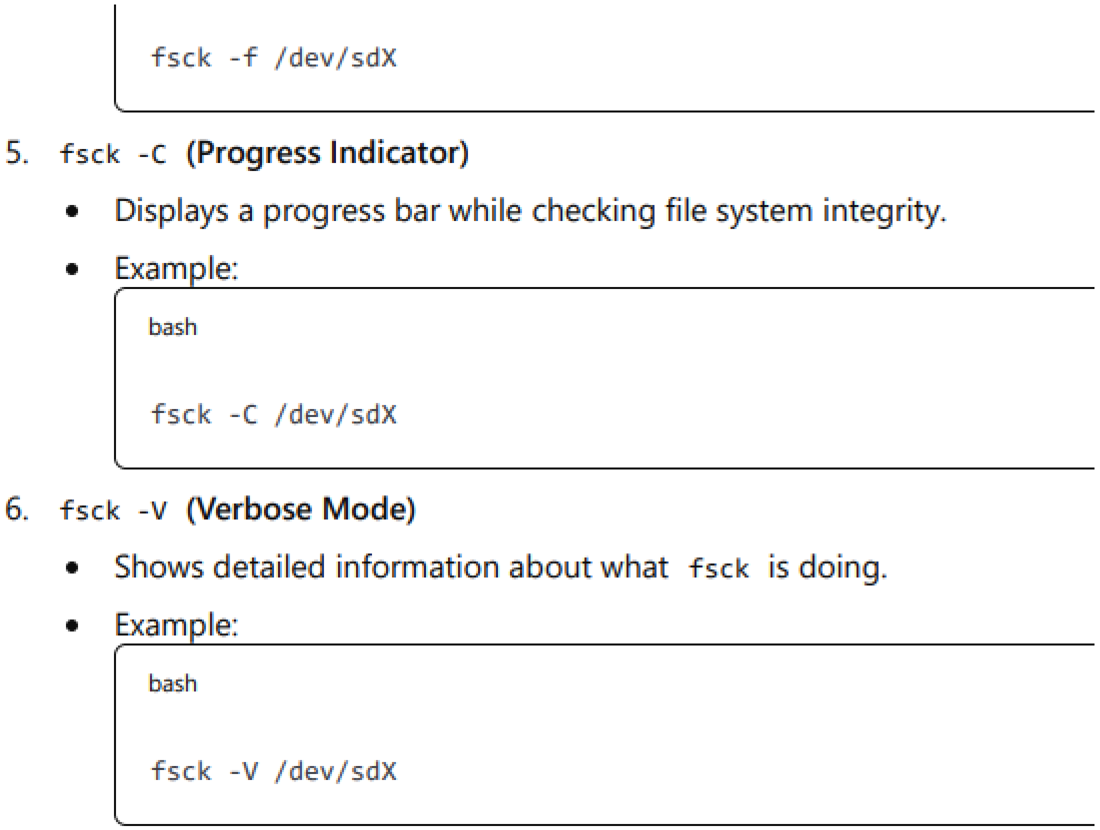
---

2. `fsck -y` **(Automatic Fixing of Errors)**
   - Automatically answers "yes" to all prompts.
   - Useful for unattended repairs.
   - Example:

     ```bash
     fsck -y /dev/sdX
     ```

3. `fsck -n` **(Read-Only Check, No Changes)**
   - Runs in read-only mode and does not modify the file system.
   - Safe to use when diagnosing issues.
   - Example:

     ```bash
     fsck -n /dev/sdX
     ```



```
fsck -f /dev/sdX
```

5. `fsck -C` **(Progress Indicator)**
   - Displays a progress bar while checking file system integrity.
   - Example:

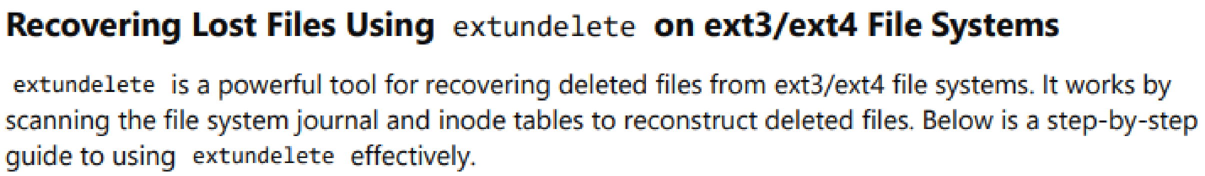
```bash
fsck -C /dev/sdX
```

6. `fsck -V` **(Verbose Mode)**
   - Shows detailed information about what `fsck` is doing.
   - Example:

```bash
fsck -V /dev/sdX
```

4.Recover Lost Files:

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



Recovering Lost Files Using `extundelete` on ext3/ext4 File Systems

`extundelete` is a powerful tool for recovering deleted files from ext3/ext4 file systems. It works by scanning the file system journal and inode tables to reconstruct deleted files. Below is a step-by-step guide to using `extundelete` effectively.

1. Install `extundelete`

Before using `extundelete`, ensure it's installed on your system. Run the following command:

bash Copy Edit

```bash
sudo apt update && sudo apt install extundelete # For Debian/Ubuntu sudo yum install
extundelete # For RHEL/CentOS
```

---

2. Unmount the Affected Partition

To maximize recovery success, **immediately stop writing to the partition** where files were deleted. Ideally, unmount the partition before proceeding.

bash Copy Edit

```bash
sudo umount /dev/sdX
```

Replace `/dev/sdX` with the actual device identifier.

3. Check the File System

Before running `extundelete`, check the integrity of the file system using `fsck`:

```bash
sudo fsck -n /dev/sdX
```

This ensures there are no major file system inconsistencies that could affect recovery.

---

4. Use `extundelete` to Scan for Recoverable Files

Run the following command to scan the partition for recoverable files:

```bash
sudo extundelete /dev/sdX --inode 2
```

5. Recover a Specific File

To recover a specific deleted file:

```bash
sudo extundelete /dev/sdX --restore-file /path/to/deleted/file
```

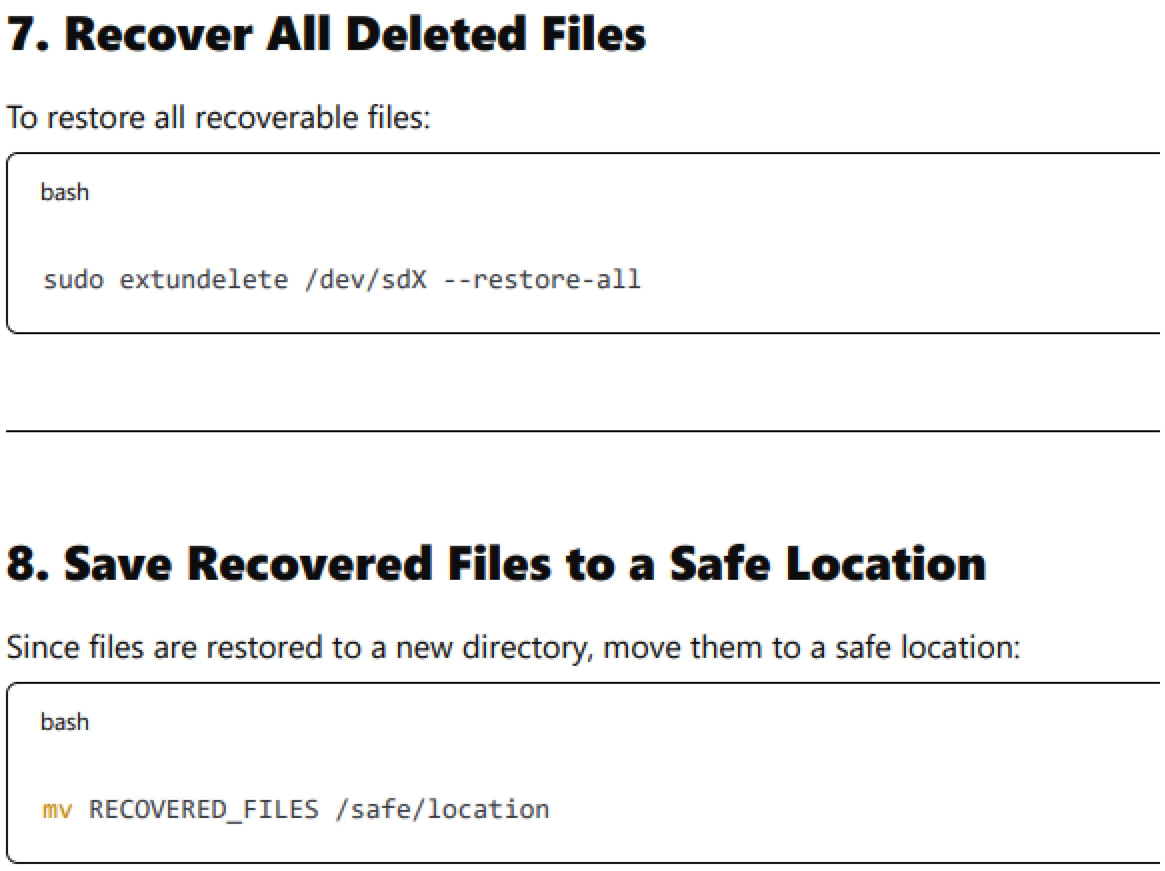
The restored file will be saved in a folder called `RECOVERED_FILES` in your working directory.

---

6. Recover an Entire Directory

To restore an entire directory:

```bash
sudo extundelete /dev/sdX --restore-directory /path/to/deleted/directory
```



7. Recover All Deleted Files

To restore all recoverable files:

```bash
sudo extundelete /dev/sdX --restore-all
```

---

8. Save Recovered Files to a Safe Location

Since files are restored to a new directory, move them to a safe location:

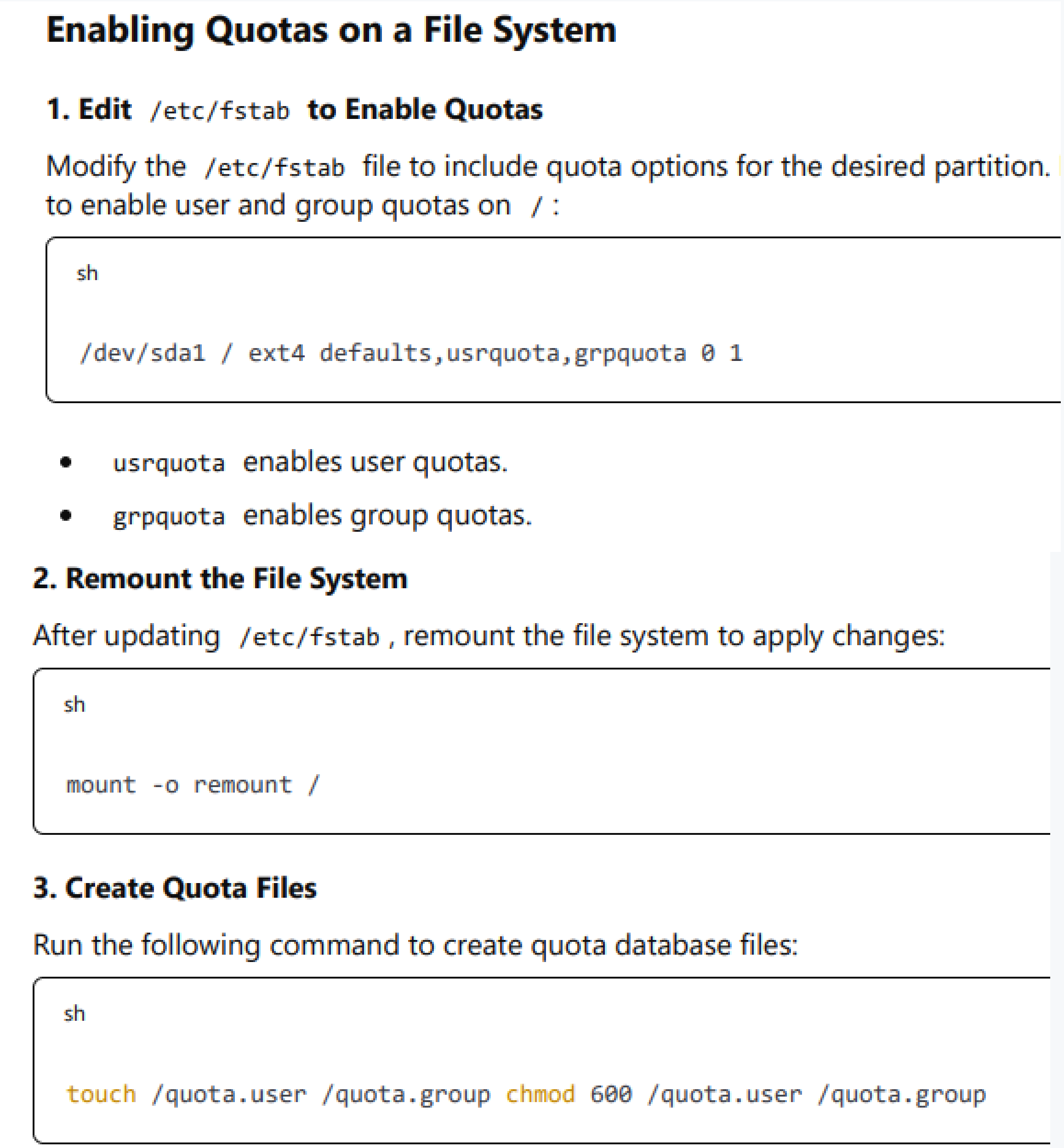
```bash
mv RECOVERED_FILES /safe/location
```

Lab 9: File System Quotas

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 /.



Enabling Quotas on a File System

1. Edit `/etc/fstab` to Enable Quotas

Modify the `/etc/fstab` file to include quota options for the desired partition.
to enable user and group quotas on `/`:

```sh
/dev/sda1 / ext4 defaults,usrquota,grpquota 0 1
```

- `usrquota` enables user quotas.
- `grpquota` enables group quotas.

2. Remount the File System

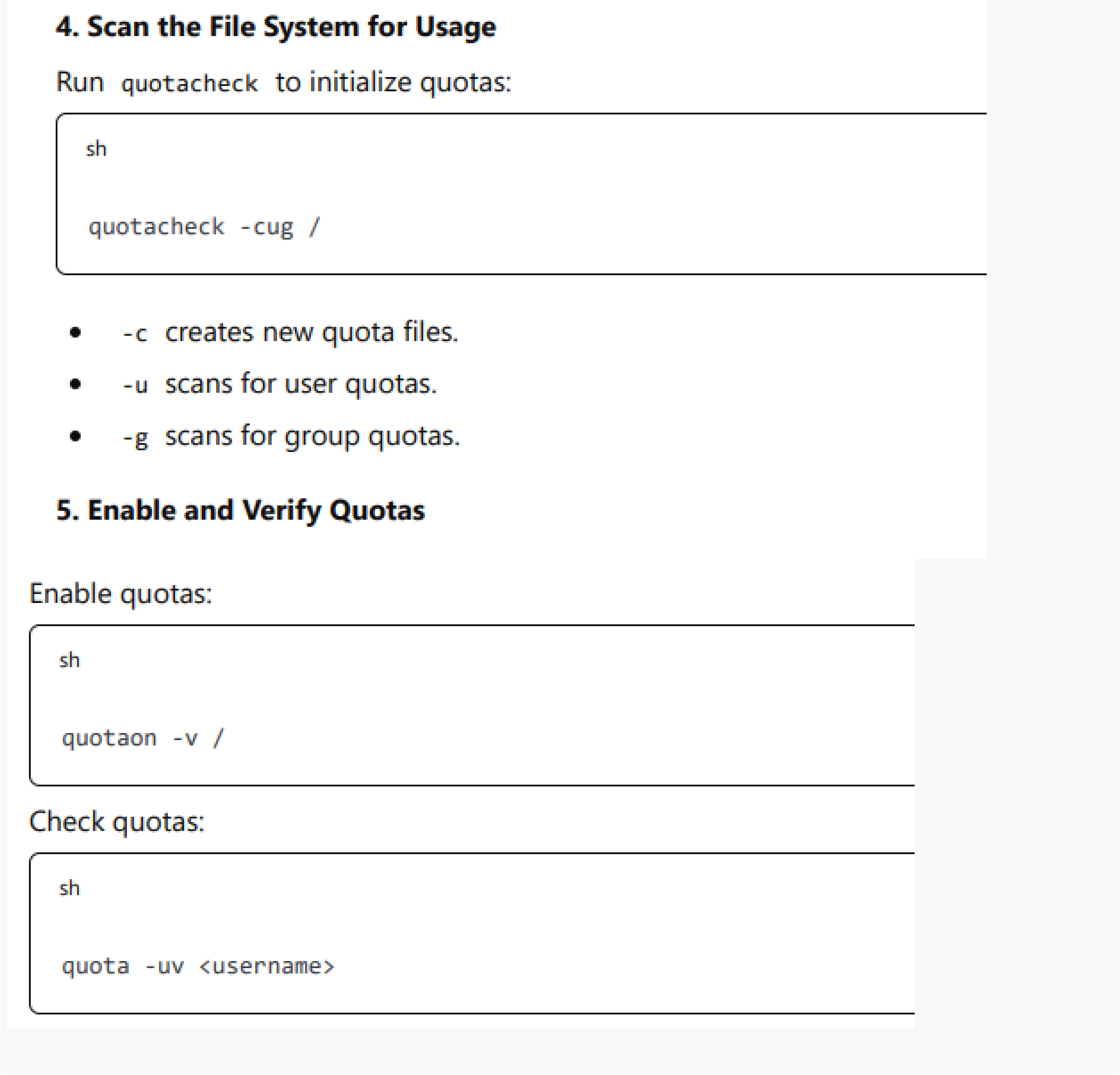
After updating `/etc/fstab`, remount the file system to apply changes:

```sh
mount -o remount /
```

3. Create Quota Files

Run the following command to create quota database files:

```sh
touch /quota.user /quota.group chmod 600 /quota.user /quota.group
```



4. Scan the File System for Usage

Run `quotacheck` to initialize quotas:

```sh
quotacheck -cug /
```

- `-c` creates new quota files.
- `-u` scans for user quotas.
- `-g` scans for group quotas.

5. Enable and Verify Quotas

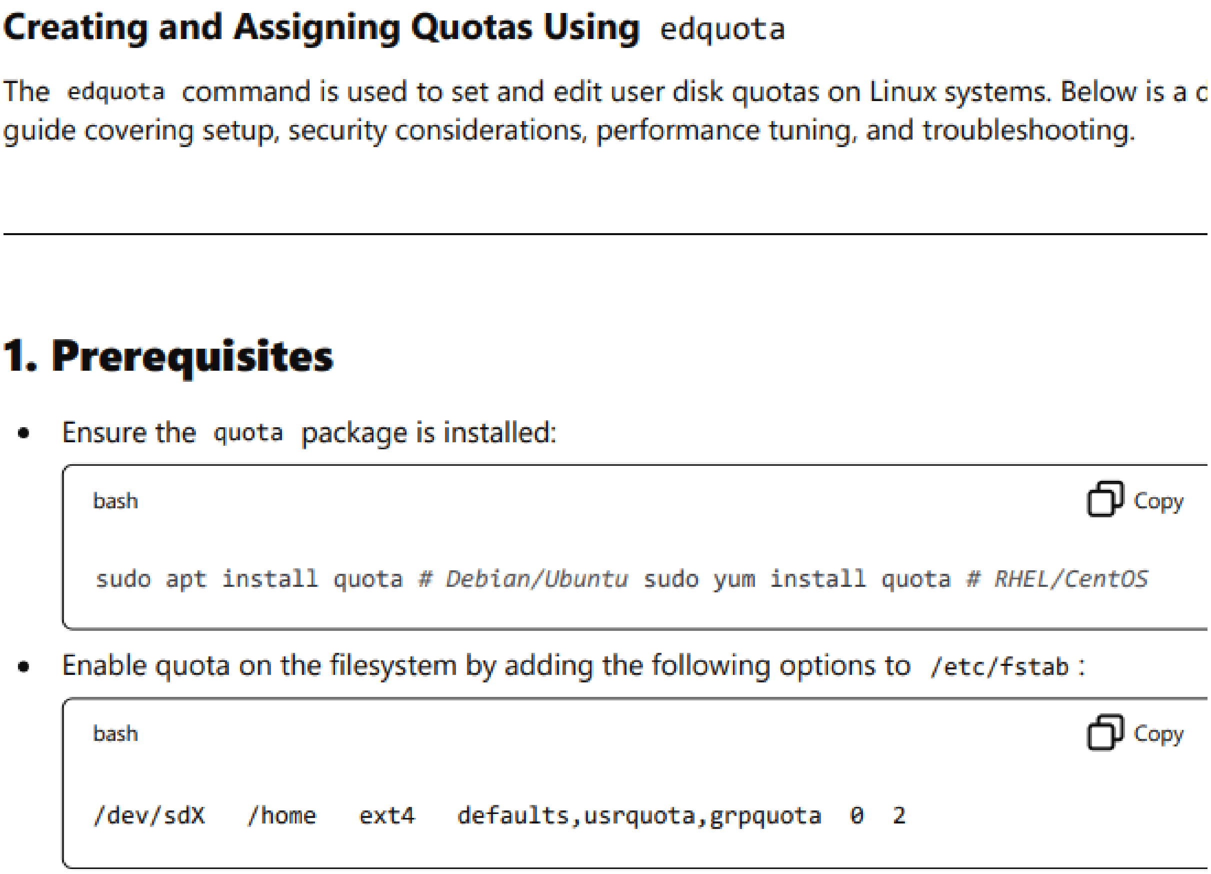
Enable quotas:

```sh
quotaon -v /
```

Check quotas:

```sh
quota -uv <username>
```

```
2.      Create and Assign Quotas:

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



Creating and Assigning Quotas Using `edquota`

The `edquota` command is used to set and edit user disk quotas on Linux systems. Below is a c guide covering setup, security considerations, performance tuning, and troubleshooting.

---

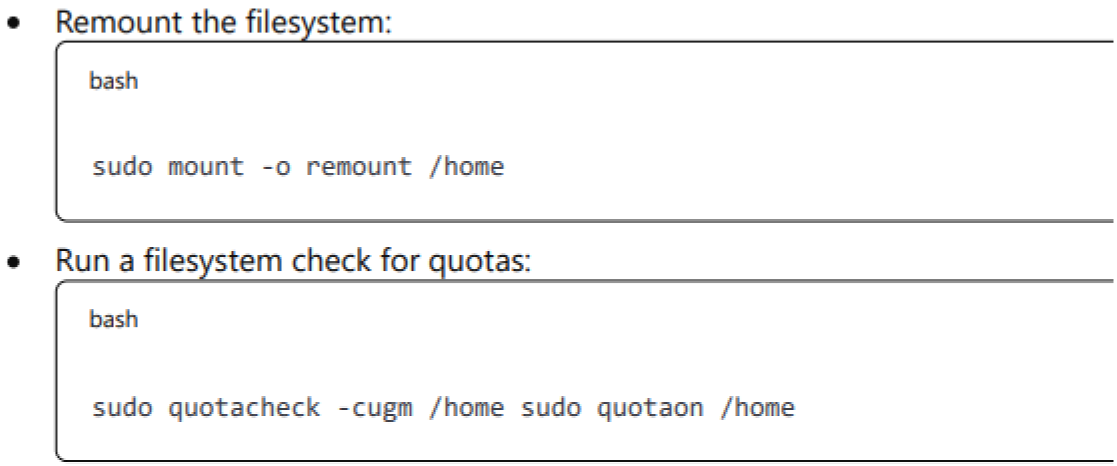
1. Prerequisites

- Ensure the `quota` package is installed:

```bash
sudo apt install quota # Debian/Ubuntu sudo yum install quota # RHEL/CentOS
```

- Enable quota on the filesystem by adding the following options to `/etc/fstab`:

```bash
/dev/sdX   /home   ext4   defaults,usrquota,grpquota  0  2
```



- Remount the filesystem:

```bash
sudo mount -o remount /home
```

- Run a filesystem check for quotas:

```bash
sudo quotacheck -cugm /home sudo quotaon /home
```

2. Setting User Quotas Using `edquota`

1. Open quota editor for a user:

bash

```bash
sudo edquota -u username
```

This opens a text editor with a format like:

bash

```bash
Disk quotas for user username (uid 1001):
  Filesystem                   blocks       soft       hard     inodes
  /dev/sda1                    50000       60000      70000       0
```

- **Soft limit**: Warning issued when exceeded (grace period applies).
- **Hard limit**: Absolute limit; user cannot exceed this.

2. Set group quotas:

bash

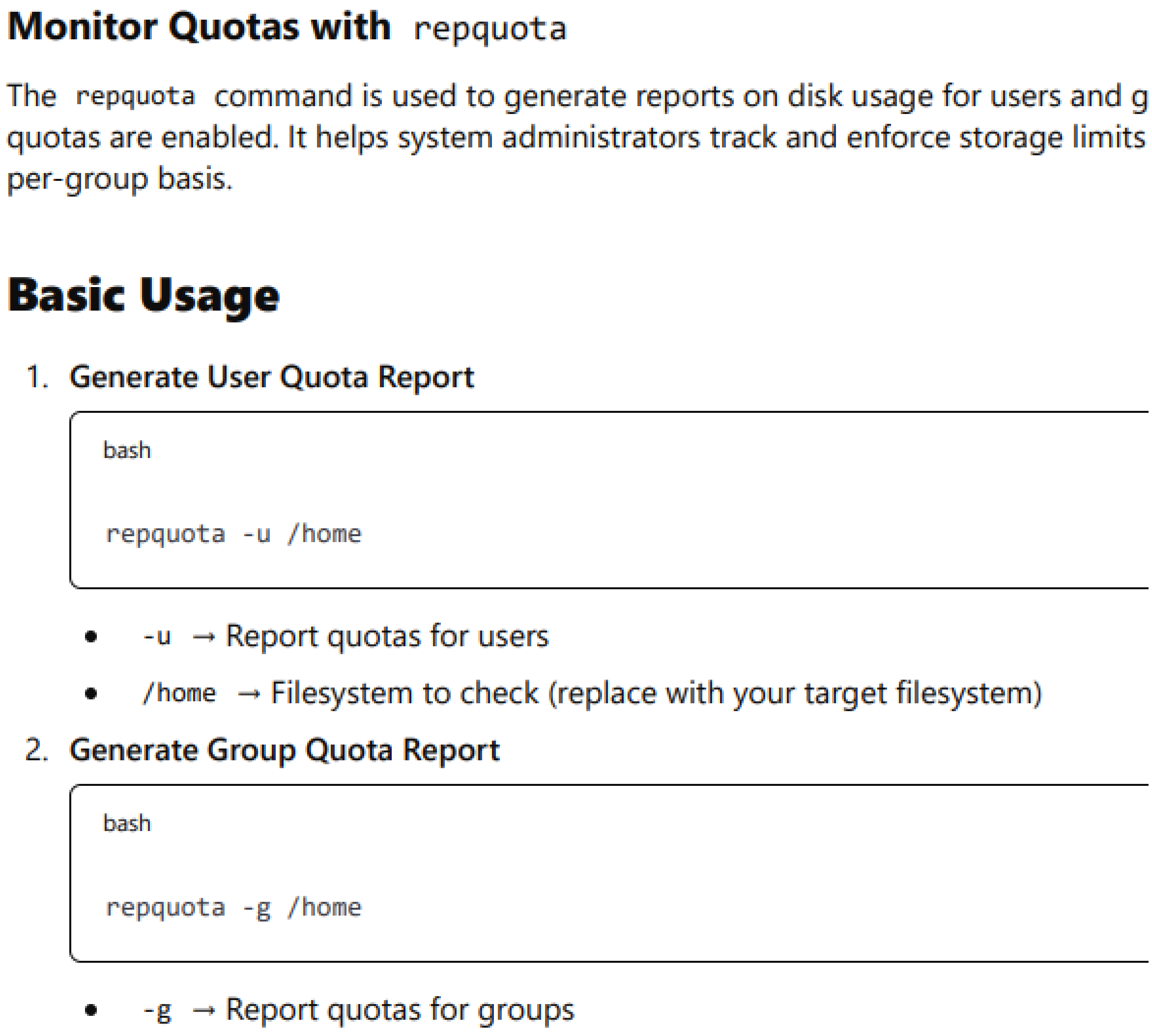
```bash
sudo edquota -g groupname
```

3. Clone quotas from one user to another:

bash

```bash
sudo edquota -p user1 user2
```

```
3.   Monitor Quotas:

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



Monitor Quotas with `repquota`

The `repquota` command is used to generate reports on disk usage for users and g
quotas are enabled. It helps system administrators track and enforce storage limits
per-group basis.

Basic Usage

1. **Generate User Quota Report**

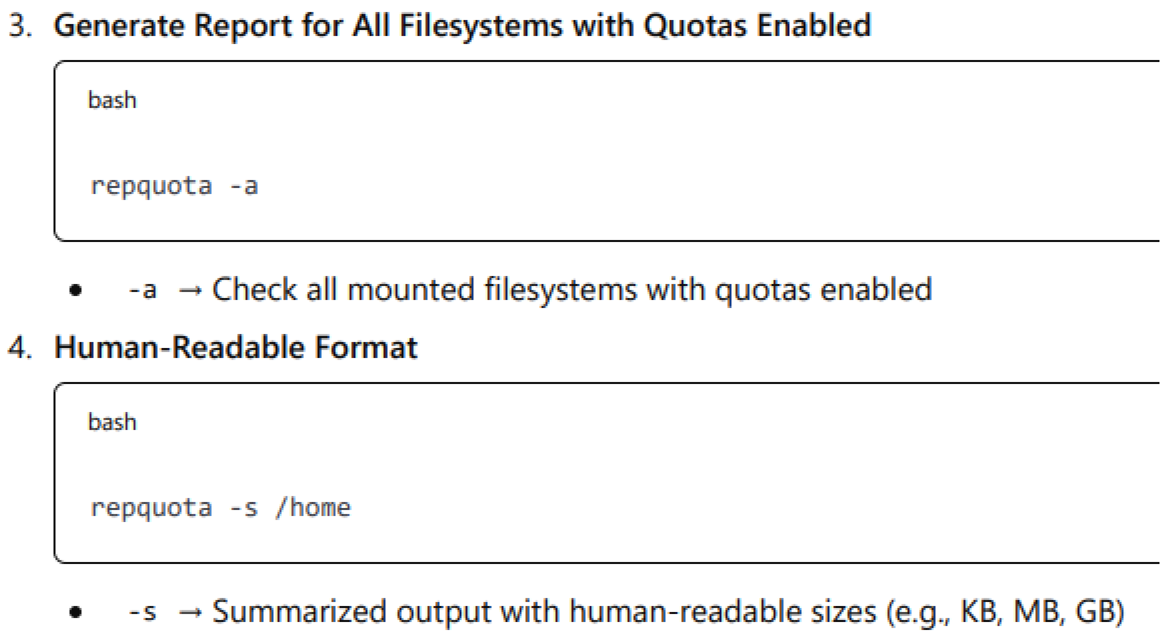
```bash
repquota -u /home
```

- `-u` → Report quotas for users
- `/home` → Filesystem to check (replace with your target filesystem)

2. **Generate Group Quota Report**

```bash
repquota -g /home
```

- `-g` → Report quotas for groups



3. **Generate Report for All Filesystems with Quotas Enabled**

```bash
repquota -a
```

- `-a` → Check all mounted filesystems with quotas enabled

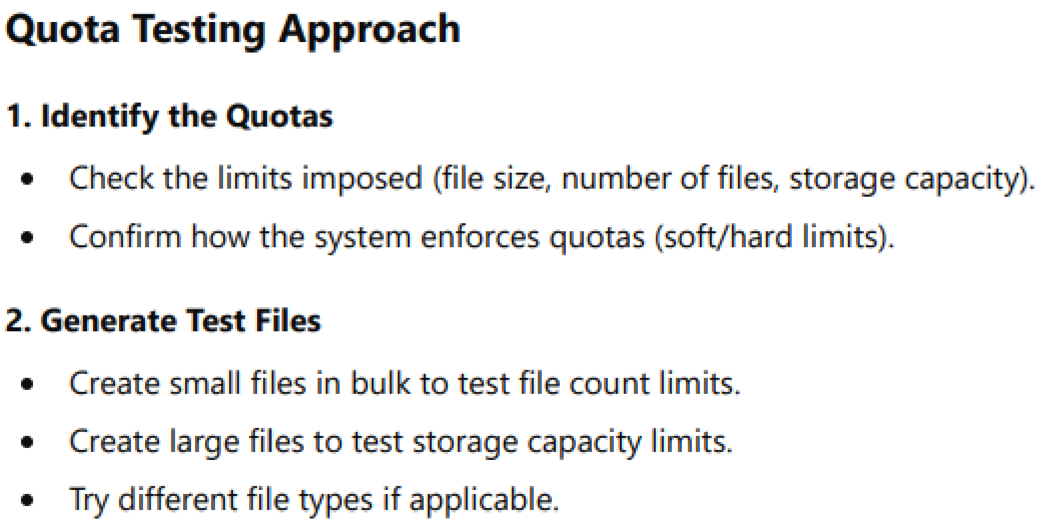
4. **Human-Readable Format**

```bash
repquota -s /home
```

- `-s` → Summarized output with human-readable sizes (e.g., KB, MB, GB)

4.Test Quotas:

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



Quota Testing Approach

1. Identify the Quotas

- Check the limits imposed (file size, number of files, storage capacity).
- Confirm how the system enforces quotas (soft/hard limits).

2. Generate Test Files

- Create small files in bulk to test file count limits.
- Create large files to test storage capacity limits.
- Try different file types if applicable.