Storage Systems and Data Management Guide 1.0 documentation

Storage Systems and Data Management Guide

Welcome to the comprehensive Storage Systems and Data Management Guide! This documentation covers everything you need to know about storage technologies, disk management, file systems, and data organization on Ubuntu 22.04 LTS.

Storage Overview

Introduction to Storage Systems

Storage systems are fundamental components of any computing environment. They provide persistent data storage, enabling programs and users to save, retrieve, and manage information efficiently.

What is Storage?

Storage refers to the various technologies and devices used to hold digital data persistently. Unlike volatile memory (RAM), storage retains data even when power is removed from the system.

Types of Storage

Primary Storage Categories

1. **Primary Storage**: Direct access storage (RAM, Cache)
2. **Secondary Storage**: Persistent storage (Hard drives, SSDs)
3. **Tertiary Storage**: Archival storage (Tape drives, Optical storage)
4. **Network Storage**: Remote storage accessed over networks

Storage Hierarchy

```
CPU Registers (fastest, smallest)
↓
Cache Memory (L1, L2, L3)
↓
Main Memory (RAM)
↓
Secondary Storage (SSD, HDD)
↓
Network Storage (NAS, SAN)
↓
Archival Storage (slowest, largest)
```

Ubuntu 22.04 Storage Commands

Basic Storage Information Commands

```
# Check disk usage
df -h

# Display detailed disk usage
du -sh /home/*

# List block devices
lsblk

# Show partition table
sudo fdisk -l

# Display mounted filesystems
mount | grep "^/dev"

# Check filesystem usage
sudo du -h --max-depth=1 /
```

Storage Performance Monitoring

```
# Monitor I/O statistics
iostat -x 1

# Watch disk activity
sudo iotop

# Check disk performance
sudo hdparm -Tt /dev/sda

# Monitor filesystem usage in real-time
watch df -h
```

Frequently Asked Questions

Q: What's the difference between storage and memory?

A: Memory (RAM) is volatile and provides temporary storage for currently running programs. Storage is non-volatile and provides permanent data storage that persists when the system is powered off.

Q: How do I check available storage space on Ubuntu 22.04?

A: Use the following commands:

```
# Human-readable format
df -h

# Show inodes usage
df -i

# Specific filesystem
df -h /home
```

Q: What storage types are best for different use cases?

A:

- **SSDs**: Best for operating systems, applications, and frequently accessed data
- **HDDs**: Ideal for bulk storage, backups, and archival data
- **NVMe SSDs**: Perfect for high-performance applications and databases
- **Network Storage**: Suitable for shared data and centralized management

Q: How can I optimize storage performance on Ubuntu?

A: Several optimization techniques:

```
# Enable TRIM for SSDs
sudo systemctl enable fstrim.timer

# Optimize filesystem for SSDs
sudo tune2fs -o discard /dev/sda1

# Mount with noatime option (in /etc/fstab)
# /dev/sda1 / ext4 defaults,noatime 0 1
```

```
# Use appropriate I/O scheduler
echo noop | sudo tee /sys/block/sda/queue/scheduler
```

Coding Examples

Python Storage Monitoring Script

```
#!/usr/bin/env python3
"""
Storage monitoring script for Ubuntu 22.04
"""
import os
import shutil
import subprocess
import json

def get_disk_usage():
    """Get disk usage information"""
    usage = shutil.disk_usage('/')
    total = usage.total
    used = usage.used
    free = usage.free

    return {
        'total_gb': round(total / (1024**3), 2),
        'used_gb': round(used / (1024**3), 2),
        'free_gb': round(free / (1024**3), 2),
        'usage_percent': round((used / total) * 100, 2)
    }

def get_mounted_filesystems():
    """Get list of mounted filesystems"""
    result = subprocess.run(['mount'], capture_output=True, text=True)
    filesystems = []

    for line in result.stdout.split('\n'):
        if line.startswith('/dev/'):
            parts = line.split()
            if len(parts) >= 3:
                filesystems.append({
                    'device': parts[0],
                    'mountpoint': parts[2],
                    'filesystem': parts[4] if len(parts) > 4 else 'unknown'
                })

    return filesystems

def monitor_io():
    """Monitor I/O statistics"""
    try:
        result = subprocess.run(['iostat', '-x', '1', '1'],
                                capture_output=True, text=True)
        return result.stdout
    except FileNotFoundError:
        return "iostat not available. Install with: sudo apt install sysstat"

if __name__ == "__main__":
    print("=== Storage System Monitor ===")
    print(f"Disk Usage: {json.dumps(get_disk_usage(), indent=2)}")
    print(f"Mounted Filesystems: {json.dumps(get_mounted_filesystems(), indent=2)}")
    print("I/O Statistics:")
    print(monitor_io())
```

Bash Storage Management Script

```
#!/bin/bash
# storage_manager.sh - Storage management utility for Ubuntu 22.04

# Colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

# Function to display disk usage
show_disk_usage() {
    echo -e "${GREEN}=== Disk Usage Information ===${NC}"
    df -h | grep -E "^/dev"
    echo ""
}

# Function to show largest directories
show_large_dirs() {
    echo -e "${GREEN}=== Largest Directories ===${NC}"
    sudo du -h /home /var /usr 2>/dev/null | sort -hr | head -10
    echo ""
}

# Function to check storage health
check_storage_health() {
    echo -e "${GREEN}=== Storage Health Check ===${NC}"

    # Check for devices
    for device in /dev/sd[a-z]; do
        if [ -b "$device" ]; then
            echo "Checking $device..."
            sudo smartctl -H "$device" 2>/dev/null || echo "SMART not available for
$device"
        fi
    done
    echo ""
}

# Function to clean temporary files
clean_temp_files() {
    echo -e "${YELLOW}=== Cleaning Temporary Files ===${NC}"

    # Clean apt cache
    sudo apt autoclean
    sudo apt autoremove

    # Clean user temp files
    rm -rf ~/.cache/thumbnails/*
    rm -rf /tmp/*

    # Clean system logs (keep last 3 days)
    sudo journalctl --vacuum-time=3d

    echo -e "${GREEN}Cleanup completed!${NC}"
}

# Main menu
case "${1:-menu}" in
    "usage")
        show_disk_usage
        ;;
    "large")
```

```
        show_large_dirs
        ;;
    "health")
        check_storage_health
        ;;
    "clean")
        clean_temp_files
        ;;
    "menu"|*)
        echo "Storage Manager for Ubuntu 22.04"
        echo "Usage: $0 [option]"
        echo ""
        echo "Options:"
        echo "  usage  - Show disk usage"
        echo "  large  - Show largest directories"
        echo "  health - Check storage health"
        echo "  clean  - Clean temporary files"
        ;;
esac
```

Best Practices

Storage Planning

1. **Capacity Planning**: Always plan for 20-30% free space
2. **Performance Requirements**: Choose storage type based on IOPS needs
3. **Redundancy**: Implement appropriate backup and RAID strategies
4. **Monitoring**: Set up alerts for storage usage thresholds

Ubuntu 22.04 Specific Recommendations

```
# Install useful storage tools
sudo apt update
sudo apt install -y htop iotop smartmontools sysstat tree ncdu

# Set up automatic TRIM for SSDs
sudo systemctl enable fstrim.timer
sudo systemctl start fstrim.timer

# Configure log rotation
sudo vim /etc/logrotate.conf

# Monitor storage usage with cron
echo "0 */6 * * * /usr/bin/df -h | mail -s 'Storage Report' admin@example.com" |
crontab -
```

Security Considerations

1. **Encryption**: Use LUKS for full disk encryption
2. **Access Controls**: Implement proper file permissions
3. **Backup Security**: Encrypt backups and test restoration
4. **Network Storage**: Use secure protocols (SSH, VPN)

```
# Set up LUKS encryption
sudo cryptsetup luksFormat /dev/sdb
```

```
sudo cryptsetup luksOpen /dev/sdb encrypted_drive

# Create filesystem on encrypted device
sudo mkfs.ext4 /dev/mapper/encrypted_drive

# Mount encrypted filesystem
sudo mkdir /mnt/encrypted
sudo mount /dev/mapper/encrypted_drive /mnt/encrypted
```

Disk Management

Understanding Disk Management

Disk management involves the organization, partitioning, formatting, and maintenance of storage devices. In Ubuntu 22.04, disk management is crucial for optimal system performance and data organization.

Disk Management Fundamentals

What is Disk Management?

Disk management encompasses:

- **Partitioning**: Dividing physical disks into logical sections
- **Formatting**: Preparing partitions with file systems
- **Mounting**: Making partitions accessible to the operating system
- **Monitoring**: Tracking disk health and performance
- **Maintenance**: Optimizing and repairing disk issues

Disk Types and Interfaces

Physical Disk Types

1. **Hard Disk Drives (HDD)**
    - Mechanical storage with spinning platters
    - Lower cost per GB
    - Higher latency, slower access times
    - Good for bulk storage and archival
2. **Solid State Drives (SSD)**
    - Flash memory-based storage
    - Faster access times, lower latency
    - Higher cost per GB
    - Ideal for operating systems and applications
3. **NVMe SSDs**

- PCIe-based interface
- Highest performance storage
- Direct CPU communication
- Best for high-performance applications

Interface Types

```
SATA (Serial ATA)
├── SATA I: 1.5 Gbps
├── SATA II: 3.0 Gbps
└── SATA III: 6.0 Gbps

NVMe (Non-Volatile Memory Express)
├── PCIe 3.0: Up to 32 Gbps
└── PCIe 4.0: Up to 64 Gbps
```

Ubuntu 22.04 Disk Management Commands

Essential Disk Commands

```
# List all block devices
lsblk

# Display partition tables
sudo fdisk -l

# Show disk usage by filesystem
df -h

# Display directory sizes
du -sh /home/*

# List mounted filesystems
mount | column -t

# Show disk I/O statistics
iostat -x 1
```

Advanced Disk Information

```
# Detailed disk information
sudo lshw -class disk

# SMART disk health information
sudo smartctl -a /dev/sda

# Disk geometry information
sudo hdparm -g /dev/sda

# Check bad blocks
sudo badblocks -v /dev/sda

# Display partition UUID
sudo blkid
```

Partitioning with fdisk

Creating Partitions

```
# Start fdisk for a specific disk
sudo fdisk /dev/sdb

# Within fdisk:
# n - Create new partition
# d - Delete partition
# p - Print partition table
# w - Write changes and exit
# q - Quit without saving

# Example: Create a new primary partition
sudo fdisk /dev/sdb << EOF
n
p
1


w
EOF
```

Partitioning with parted

```
# Create GPT partition table
sudo parted /dev/sdb mklabel gpt

# Create partition
sudo parted /dev/sdb mkpart primary ext4 0% 100%

# Set partition flags
sudo parted /dev/sdb set 1 boot on

# Display partition information
sudo parted /dev/sdb print
```

Filesystem Creation and Management

Creating Filesystems

```
# Create ext4 filesystem
sudo mkfs.ext4 /dev/sdb1

# Create XFS filesystem
sudo mkfs.xfs /dev/sdb1

# Create Btrfs filesystem
sudo mkfs.btrfs /dev/sdb1

# Create FAT32 filesystem
sudo mkfs.fat -F32 /dev/sdb1
```

Filesystem Checking and Repair

```
# Check ext4 filesystem
sudo fsck.ext4 /dev/sdb1

# Force check even if clean
sudo fsck.ext4 -f /dev/sdb1

# Check and repair filesystem
sudo fsck.ext4 -p /dev/sdb1

# Check XFS filesystem
```

```
sudo xfs_check /dev/sdb1
```

Mounting and Unmounting

Manual Mounting

```
# Create mount point
sudo mkdir /mnt/mydisk

# Mount filesystem
sudo mount /dev/sdb1 /mnt/mydisk

# Mount with specific options
sudo mount -o rw,noatime /dev/sdb1 /mnt/mydisk

# Unmount filesystem
sudo umount /mnt/mydisk

# Force unmount (use carefully)
sudo umount -f /mnt/mydisk
```

Automatic Mounting with /etc/fstab

```
# Edit fstab file
sudo vim /etc/fstab

# Example fstab entries:
# UUID=12345678-1234-1234-1234-123456789012 /mnt/mydisk ext4 defaults 0 2
# /dev/sdb1 /mnt/backup ext4 rw,noatime 0 2

# Get UUID of a partition
sudo blkid /dev/sdb1

# Test fstab configuration
sudo mount -a
```

Frequently Asked Questions

Q: How do I check disk space usage on Ubuntu 22.04?

A: Use these commands to check disk space:

```
# Overall disk usage
df -h

# Specific directory usage
du -sh /home/username

# Interactive disk usage browser
sudo apt install ncdu
ncdu /
```

Q: What's the difference between fdisk and parted?

A:

- **fdisk**: Traditional partitioning tool, best for MBR partitions
- **parted**: Modern tool supporting both MBR and GPT, better for large disks (>2TB)

```
# fdisk - good for disks < 2TB
sudo fdisk /dev/sdb

# parted - better for large disks and GPT
sudo parted /dev/sdb
```

Q: How do I resize a partition without losing data?

A: Use these steps carefully:

```
# 1. Backup your data first!
# 2. Unmount the partition
sudo umount /dev/sdb1

# 3. Check filesystem
sudo fsck -f /dev/sdb1

# 4. Resize partition with parted
sudo parted /dev/sdb resizepart 1 100%

# 5. Resize filesystem
sudo resize2fs /dev/sdb1

# 6. Remount
sudo mount /dev/sdb1 /mnt/mydisk
```

Q: How can I securely wipe a disk?

A: Several methods for secure disk wiping:

```
# Method 1: Using dd (single pass)
sudo dd if=/dev/zero of=/dev/sdb bs=1M status=progress

# Method 2: Using shred (multiple passes)
sudo shred -vfz -n 3 /dev/sdb

# Method 3: Using DBAN (boot from USB)
# Download DBAN ISO and create bootable USB

# For SSDs, use secure erase
sudo hdparm --user-master u --security-set-pass p /dev/sdb
sudo hdparm --user-master u --security-erase p /dev/sdb
```

Coding Examples

Python Disk Management Script

```
#!/usr/bin/env python3
"""
Disk management utility for Ubuntu 22.04
"""
import subprocess
import json
import os
import sys

class DiskManager:
    def __init__(self):
        self.check_root_privileges()

    def check_root_privileges(self):
        """Check if running with root privileges"""
```

```
        if os.geteuid() != 0:
            print("This script requires root privileges. Run with sudo.")
            sys.exit(1)

    def list_disks(self):
        """List all available disks"""
        try:
            result = subprocess.run(['lsblk', '-J'], capture_output=True, text=True)
            data = json.loads(result.stdout)
            return data['blockdevices']
        except Exception as e:
            print(f"Error listing disks: {e}")
            return []

    def get_disk_info(self, device):
        """Get detailed information about a disk"""
        try:
            # Get basic info
            result = subprocess.run(['lsblk', '-J', device],
                                    capture_output=True, text=True)
            basic_info = json.loads(result.stdout)

            # Get SMART info
            smart_result = subprocess.run(['smartctl', '-i', device],
                                          capture_output=True, text=True)

            return {
                'basic': basic_info,
                'smart': smart_result.stdout if smart_result.returncode == 0 else
'N/A'
            }
        except Exception as e:
            return {'error': str(e)}

    def create_partition(self, device, size='100%'):
        """Create a new partition"""
        try:
            # Create GPT partition table
            subprocess.run(['parted', device, 'mklabel', 'gpt'], check=True)

            # Create partition
            subprocess.run(['parted', device, 'mkpart', 'primary',
                            'ext4', '0%', size], check=True)

            return True
        except subprocess.CalledProcessError as e:
            print(f"Error creating partition: {e}")
            return False

    def format_partition(self, device, filesystem='ext4'):
        """Format a partition with specified filesystem"""
        try:
            if filesystem == 'ext4':
                subprocess.run(['mkfs.ext4', '-F', device], check=True)
            elif filesystem == 'xfs':
                subprocess.run(['mkfs.xfs', '-f', device], check=True)
            elif filesystem == 'btrfs':
                subprocess.run(['mkfs.btrfs', '-f', device], check=True)
            else:
                raise ValueError(f"Unsupported filesystem: {filesystem}")

            return True
        except subprocess.CalledProcessError as e:
            print(f"Error formatting partition: {e}")
            return False
```

```
    def mount_partition(self, device, mountpoint, options='defaults'):
        """Mount a partition"""
        try:
            # Create mountpoint if it doesn't exist
            os.makedirs(mountpoint, exist_ok=True)

            # Mount the partition
            subprocess.run(['mount', '-o', options, device, mountpoint],
                           check=True)

            return True
        except subprocess.CalledProcessError as e:
            print(f"Error mounting partition: {e}")
            return False

    def check_filesystem(self, device):
        """Check filesystem integrity"""
        try:
            result = subprocess.run(['fsck', '-n', device],
                                    capture_output=True, text=True)
            return {
                'status': 'clean' if result.returncode == 0 else 'errors',
                'output': result.stdout
            }
        except subprocess.CalledProcessError as e:
            return {'status': 'error', 'output': str(e)}

# Example usage
if __name__ == "__main__":
    dm = DiskManager()

    print("=== Available Disks ===")
    disks = dm.list_disks()
    for disk in disks:
        print(f"Device: {disk['name']}, Size: {disk['size']}, Type: {disk['type']}")

    # Interactive disk management
    if len(sys.argv) > 1:
        device = sys.argv[1]
        info = dm.get_disk_info(device)
        print(f"\\nDisk information for {device}:")
        print(json.dumps(info, indent=2))
```

Bash Disk Monitoring Script

```
#!/bin/bash
# disk_monitor.sh - Comprehensive disk monitoring for Ubuntu 22.04

# Configuration
ALERT_THRESHOLD=85  # Alert when disk usage exceeds this percentage
EMAIL_ALERT="admin@example.com"
LOG_FILE="/var/log/disk_monitor.log"

# Colors
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m'

# Logging function
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$LOG_FILE"
```

```
}

# Check disk usage
check_disk_usage() {
    echo -e "${BLUE}=== Disk Usage Check ===${NC}"

    df -h | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{print $5 " " $1 " " $6}' |
while read output; do
        usage=$(echo $output | awk '{print $1}' | sed 's/%//g')
        partition=$(echo $output | awk '{print $2}')
        mountpoint=$(echo $output | awk '{print $3}')

        if [ $usage -ge $ALERT_THRESHOLD ]; then
            echo -e "${RED}WARNING: $partition ($mountpoint) is ${usage}% full${NC}"
            log_message "HIGH USAGE: $partition ($mountpoint) is ${usage}% full"

            # Send email alert if configured
            if command -v mail &> /dev/null && [ ! -z "$EMAIL_ALERT" ]; then
                echo "Disk usage alert: $partition ($mountpoint) is ${usage}% full" |
\
                    mail -s "Disk Usage Alert - $(hostname)" "$EMAIL_ALERT"
            fi
        else
            echo -e "${GREEN}OK: $partition ($mountpoint) is ${usage}% full${NC}"
        fi
    done
}

# Check disk health using SMART
check_disk_health() {
    echo -e "${BLUE}=== Disk Health Check ===${NC}"

    for disk in /dev/sd[a-z]; do
        if [ -b "$disk" ]; then
            echo "Checking $disk..."

            if command -v smartctl &> /dev/null; then
                health=$(smartctl -H "$disk" 2>/dev/null | grep "SMART overall-health"
| awk '{print $6}')

                if [ "$health" = "PASSED" ]; then
                    echo -e "${GREEN}$disk: Health OK${NC}"
                else
                    echo -e "${RED}$disk: Health FAILED${NC}"
                    log_message "DISK HEALTH: $disk health check failed"
                fi

                # Check temperature
                temp=$(smartctl -A "$disk" 2>/dev/null | grep Temperature_Celsius |
awk '{print $10}')
                if [ ! -z "$temp" ]; then
                    if [ "$temp" -gt 55 ]; then
                        echo -e "${YELLOW}$disk: Temperature high (${temp}°C)${NC}"
                    else
                        echo -e "${GREEN}$disk: Temperature OK (${temp}°C)${NC}"
                    fi
                fi
            else
                echo "smartctl not available. Install with: sudo apt install
smartmontools"
            fi
        fi
    done
}
```

```
# Monitor I/O performance
monitor_io() {
    echo -e "${BLUE}=== I/O Performance Monitor ===${NC}"

    if command -v iostat &> /dev/null; then
        iostat -x 1 3 | tail -n +4
    else
        echo "iostat not available. Install with: sudo apt install sysstat"
    fi
}

# Find large files
find_large_files() {
    echo -e "${BLUE}=== Large Files (>1GB) ===${NC}"

    find / -type f -size +1G -exec ls -lh {} \; 2>/dev/null | \
        awk '{print $5 "\t" $9}' | sort -hr | head -10
}

# Clean temporary files
cleanup_temp() {
    echo -e "${BLUE}=== Cleaning Temporary Files ===${NC}"

    # Get initial disk usage
    initial_usage=$(df / | tail -1 | awk '{print $3}')

    # Clean apt cache
    apt-get autoclean -y
    apt-get autoremove -y

    # Clean logs older than 30 days
    find /var/log -name "*.log" -mtime +30 -delete

    # Clean temporary files
    find /tmp -type f -mtime +7 -delete
    find /var/tmp -type f -mtime +7 -delete

    # Clean user caches
    find /home -name ".cache" -type d -exec rm -rf {}/thumbnails/* \; 2>/dev/null

    # Get final disk usage
    final_usage=$(df / | tail -1 | awk '{print $3}')

    # Calculate space freed
    space_freed=$((initial_usage - final_usage))
    echo -e "${GREEN}Cleanup completed. Space freed: ${space_freed}KB${NC}"

    log_message "CLEANUP: Freed ${space_freed}KB of disk space"
}

# Generate disk report
generate_report() {
    report_file="/tmp/disk_report_$(date +%Y%m%d_%H%M%S).txt"

    {
        echo "Disk Report for $(hostname) - $(date)"
        echo "================================================"
        echo ""

        echo "Disk Usage:"
        df -h
        echo ""

        echo "Largest Directories:"
        du -h /home /var /usr 2>/dev/null | sort -hr | head -10
```

```
        echo ""

        echo "Mount Points:"
        mount | grep "^/dev"
        echo ""

        echo "Block Devices:"
        lsblk

    } > "$report_file"

    echo -e "${GREEN}Report generated: $report_file${NC}"

    # Email report if configured
    if command -v mail &> /dev/null && [ ! -z "$EMAIL_ALERT" ]; then
        cat "$report_file" | mail -s "Disk Report - $(hostname)" "$EMAIL_ALERT"
    fi
}

# Main menu
case "${1:-help}" in
    "usage")
        check_disk_usage
        ;;
    "health")
        check_disk_health
        ;;
    "io")
        monitor_io
        ;;
    "large")
        find_large_files
        ;;
    "clean")
        cleanup_temp
        ;;
    "report")
        generate_report
        ;;
    "all")
        check_disk_usage
        check_disk_health
        monitor_io
        find_large_files
        ;;
    "help"|*)
        echo "Disk Monitor for Ubuntu 22.04"
        echo "Usage: $0 [option]"
        echo ""
        echo "Options:"
        echo "  usage  - Check disk usage"
        echo "  health - Check disk health (SMART)"
        echo "  io     - Monitor I/O performance"
        echo "  large  - Find large files"
        echo "  clean  - Clean temporary files"
        echo "  report - Generate comprehensive report"
        echo "  all    - Run all checks"
        ;;
esac
```

Best Practices for Disk Management

Planning and Strategy

1. **Partition Strategy**:
    - Separate /home from root partition
    - Use dedicated partitions for /var and /tmp
    - Plan for future expansion
2. **Filesystem Selection**:
    - ext4: General purpose, reliable
    - XFS: Large files and high performance
    - Btrfs: Advanced features, snapshots
    - ZFS: Enterprise features, data integrity
3. **Monitoring and Maintenance**:
    - Regular SMART checks
    - Monitor disk usage trends
    - Schedule regular filesystem checks
    - Implement log rotation

Ubuntu 22.04 Optimization Tips

```
# Enable automatic TRIM for SSDs
sudo systemctl enable fstrim.timer

# Optimize mount options for performance
# Add to /etc/fstab:
# /dev/sda1 / ext4 defaults,noatime,discard 0 1

# Configure I/O scheduler for SSDs
echo none | sudo tee /sys/block/sda/queue/scheduler

# Set up monitoring
sudo apt install smartmontools sysstat
sudo systemctl enable smartd

# Configure SMART monitoring
sudo vim /etc/smartd.conf
# Add: /dev/sda -a -o on -S on -s (S/../../.02|L/../../6/03)
```

Security and Recovery

1. **Backup Strategy**:
    - Regular automated backups
    - Test restoration procedures
    - Offsite backup storage
    - Document recovery procedures
2. **Access Control**:

    - Proper file permissions
    - User and group management
    - Audit trail logging
    - Encryption for sensitive data

```
# Set up automatic backups
sudo apt install rsync

# Create backup script
cat > /usr/local/bin/backup.sh << 'EOF'
#!/bin/bash
rsync -avz --delete /home/ /backup/home/
rsync -avz --delete /etc/ /backup/etc/
EOF

chmod +x /usr/local/bin/backup.sh

# Schedule with cron
echo "0 2 * * * /usr/local/bin/backup.sh" | sudo crontab -
```

File Systems

Understanding File Systems

A file system is a method used by operating systems to store, organize, and manage files on storage devices. It defines how data is stored, accessed, and organized on disks.

What is a File System?

A file system provides:

- **File Organization**: Hierarchical structure for organizing files and directories
- **Metadata Management**: Information about files (size, permissions, timestamps)
- **Space Management**: Allocation and deallocation of disk space
- **Access Control**: Security and permission mechanisms
- **Data Integrity**: Protection against data corruption

File System Components

Core Components

1. **Superblock**: Contains metadata about the filesystem
2. **Inode Table**: Stores file metadata and pointers to data blocks
3. **Data Blocks**: Actual file content storage
4. **Directory Structure**: Hierarchical organization of files and folders

5. **Journal**: Transaction log for filesystem changes (in journaling filesystems)

File System Structure

```
Filesystem Layout
├── Superblock (filesystem metadata)
├── Group Descriptors (block group information)
├── Block Bitmap (free/used block tracking)
├── Inode Bitmap (free/used inode tracking)
├── Inode Table (file metadata)
└── Data Blocks (actual file content)
```

Common File Systems in Ubuntu 22.04

ext4 (Fourth Extended Filesystem)

The default filesystem for Ubuntu 22.04:

Features: * Journaling for data integrity * Large file and filesystem support (up to 1 EB) * Backward compatibility with ext2/ext3 * Online defragmentation * Delayed allocation

Use Cases: * General purpose computing * Desktop and server installations * Boot partitions * Home directories

```
# Create ext4 filesystem
sudo mkfs.ext4 /dev/sdb1

# Check ext4 filesystem
sudo fsck.ext4 /dev/sdb1

# Get ext4 filesystem information
sudo tune2fs -l /dev/sdb1

# Optimize ext4 filesystem
sudo tune2fs -o journal_data_writeback /dev/sdb1
```

XFS (eXtended File System)

High-performance 64-bit journaling filesystem:

Features: * Excellent scalability * Online resizing (grow only) * Advanced quota management * Allocation groups for parallel I/O * Metadata journaling

Use Cases: * Large files and databases * High-performance computing * Video editing and media storage * Enterprise storage systems

```
# Create XFS filesystem
sudo mkfs.xfs /dev/sdb1

# Check XFS filesystem
sudo xfs_check /dev/sdb1

# Repair XFS filesystem
sudo xfs_repair /dev/sdb1

# Get XFS information
sudo xfs_info /dev/sdb1

# Resize XFS filesystem (grow only)
sudo xfs_growfs /mnt/xfs
```

Btrfs (B-tree File System)

Modern copy-on-write filesystem:

Features: * Snapshots and cloning * Built-in RAID support * Compression (zlib, lzo, zstd) * Checksumming for data integrity * Online resizing (grow and shrink)

Use Cases: * System snapshots * Development environments * Data deduplication scenarios * Advanced storage management

```
# Create Btrfs filesystem
sudo mkfs.btrfs /dev/sdb1

# Mount with compression
sudo mount -o compress=zstd /dev/sdb1 /mnt/btrfs

# Create snapshot
sudo btrfs subvolume snapshot /mnt/btrfs /mnt/btrfs/snapshot

# List subvolumes
sudo btrfs subvolume list /mnt/btrfs
```

ZFS (Zettabyte File System)

Advanced filesystem with built-in volume management:

Features: * Built-in RAID (RAID-Z) * Snapshots and clones * Data deduplication * Compression * End-to-end checksumming

Installation and Use:

```
# Install ZFS on Ubuntu 22.04
sudo apt update
sudo apt install zfsutils-linux

# Create ZFS pool
sudo zpool create mypool /dev/sdb

# Create ZFS dataset
sudo zfs create mypool/data

# Enable compression
sudo zfs set compression=lz4 mypool/data

# Create snapshot
sudo zfs snapshot mypool/data@snapshot1
```

FAT32 and NTFS

FAT32: Universal compatibility, limited file size (4GB max) **NTFS**: Windows filesystem with advanced features

```
# Create FAT32 filesystem
sudo mkfs.fat -F32 /dev/sdb1

# Create NTFS filesystem
sudo mkfs.ntfs /dev/sdb1

# Mount NTFS with full permissions
sudo mount -t ntfs-3g /dev/sdb1 /mnt/ntfs -o permissions
```

File System Operations in Ubuntu 22.04

Creating File Systems

```
# ext4 with custom options
sudo mkfs.ext4 -L "MyData" -m 1 /dev/sdb1

# XFS with custom block size
sudo mkfs.xfs -f -b size=4096 /dev/sdb1

# Btrfs with multiple devices (RAID)
sudo mkfs.btrfs -d raid1 -m raid1 /dev/sdb1 /dev/sdc1

# Set filesystem label
sudo e2label /dev/sdb1 "DataDisk"
```

Mounting File Systems

```
# Basic mounting
sudo mount /dev/sdb1 /mnt/data

# Mount with specific options
sudo mount -o rw,noatime,discard /dev/sdb1 /mnt/data

# Mount by UUID (preferred method)
sudo mount UUID=12345678-1234-1234-1234-123456789abc /mnt/data

# Mount with user permissions
sudo mount -o uid=1000,gid=1000 /dev/sdb1 /mnt/data
```

Checking and Repairing File Systems

```
# Check filesystem (read-only)
sudo fsck -n /dev/sdb1

# Automatic repair
sudo fsck -p /dev/sdb1

# Force check and repair
sudo fsck -f /dev/sdb1

# Check specific filesystem types
sudo fsck.ext4 /dev/sdb1
sudo xfs_check /dev/sdb1
sudo btrfs check /dev/sdb1
```

File System Monitoring

Monitoring Tools and Commands

```
# Real-time filesystem usage
watch df -h

# Inode usage
df -i

# Detailed filesystem information
stat -f /

# Monitor filesystem I/O
sudo iotop -o

# Check filesystem fragmentation (ext4)
sudo e4defrag -c /dev/sdb1
```

```
# Btrfs filesystem usage
sudo btrfs filesystem usage /mnt/btrfs
```

Frequently Asked Questions

Q: Which filesystem should I choose for Ubuntu 22.04?

A: Filesystem selection depends on your use case:

- **ext4**: Best general-purpose choice, default for Ubuntu
- **XFS**: Large files, databases, high-performance needs
- **Btrfs**: Need snapshots, compression, or advanced features
- **ZFS**: Enterprise features, data integrity critical

```
# For most users (recommended)
sudo mkfs.ext4 /dev/sdb1

# For large files and databases
sudo mkfs.xfs /dev/sdb1

# For snapshots and modern features
sudo mkfs.btrfs /dev/sdb1
```

Q: How do I convert between filesystems?

A: Filesystem conversion requires backup and restore:

```
# 1. Backup data
sudo rsync -av /mnt/source/ /backup/

# 2. Unmount and recreate filesystem
sudo umount /mnt/source
sudo mkfs.ext4 /dev/sdb1

# 3. Mount and restore data
sudo mount /dev/sdb1 /mnt/source
sudo rsync -av /backup/ /mnt/source/
```

Q: How do I optimize filesystem performance?

A: Several optimization techniques:

```
# For SSDs - enable TRIM
sudo mount -o discard /dev/sdb1 /mnt/data

# Disable access time updates
sudo mount -o noatime /dev/sdb1 /mnt/data

# For databases - use direct I/O
sudo mount -o barrier=0 /dev/sdb1 /mnt/database

# Optimize ext4 for SSDs
sudo tune2fs -o discard /dev/sdb1
```

Q: How do I recover deleted files?

A: Recovery methods depend on the filesystem:

```
# Install recovery tools
sudo apt install testdisk photorec extundelete

# For ext4 filesystems
sudo extundelete /dev/sdb1 --restore-all

# General purpose recovery
sudo photorec /dev/sdb1

# For immediate action after deletion
sudo grep -a -B25 -A25 'text from deleted file' /dev/sdb1
```

Coding Examples

Python Filesystem Analyzer

```
#!/usr/bin/env python3
"""
Filesystem analyzer for Ubuntu 22.04
"""
import os
import subprocess
import json
import time
from pathlib import Path

class FilesystemAnalyzer:
    def __init__(self):
        self.mountpoints = self.get_mountpoints()

    def get_mountpoints(self):
        """Get all mounted filesystems"""
        mountpoints = []
        try:
            with open('/proc/mounts', 'r') as f:
                for line in f:
                    parts = line.strip().split()
                    if len(parts) >= 3 and parts[0].startswith('/dev/'):
                        mountpoints.append({
                            'device': parts[0],
                            'mountpoint': parts[1],
                            'filesystem': parts[2],
                            'options': parts[3]
                        })
        except Exception as e:
            print(f"Error reading mount points: {e}")

        return mountpoints

    def analyze_filesystem(self, path='/'):
        """Analyze filesystem usage and characteristics"""
        try:
            # Get basic filesystem statistics
            statvfs = os.statvfs(path)

            total_size = statvfs.f_frsize * statvfs.f_blocks
            free_size = statvfs.f_frsize * statvfs.f_bavail
            used_size = total_size - free_size

            # Get inode information
            total_inodes = statvfs.f_files
            free_inodes = statvfs.f_favail
            used_inodes = total_inodes - free_inodes
```

```
            return {
                'path': path,
                'total_size_gb': round(total_size / (1024**3), 2),
                'used_size_gb': round(used_size / (1024**3), 2),
                'free_size_gb': round(free_size / (1024**3), 2),
                'usage_percent': round((used_size / total_size) * 100, 2),
                'total_inodes': total_inodes,
                'used_inodes': used_inodes,
                'free_inodes': free_inodes,
                'inode_usage_percent': round((used_inodes / total_inodes) * 100, 2) if
total_inodes > 0 else 0
            }
        except Exception as e:
            return {'error': str(e)}

    def get_largest_files(self, path='/', min_size_mb=100, count=10):
        """Find largest files in a directory tree"""
        large_files = []
        min_size_bytes = min_size_mb * 1024 * 1024

        try:
            for root, dirs, files in os.walk(path):
                # Skip certain directories to avoid permission errors
                dirs[:] = [d for d in dirs if not d.startswith('.') and d not in
['proc', 'sys', 'dev']]

                for file in files:
                    try:
                        file_path = os.path.join(root, file)
                        file_size = os.path.getsize(file_path)

                        if file_size >= min_size_bytes:
                            large_files.append({
                                'path': file_path,
                                'size_mb': round(file_size / (1024**2), 2),
                                'size_gb': round(file_size / (1024**3), 2)
                            })
                    except (OSError, IOError):
                        continue  # Skip files we can't access

            # Sort by size and return top files
            large_files.sort(key=lambda x: x['size_mb'], reverse=True)
            return large_files[:count]

        except Exception as e:
            return [{'error': str(e)}]

    def get_directory_sizes(self, path='/', max_depth=2):
        """Get sizes of directories"""
        directory_sizes = []

        try:
            result = subprocess.run(['du', '-h', f'--max-depth={max_depth}', path],
                                    capture_output=True, text=True)

            for line in result.stdout.strip().split('\\n'):
                if line:
                    parts = line.split('\\t')
                    if len(parts) == 2:
                        size, dir_path = parts
                        directory_sizes.append({
                            'path': dir_path,
                            'size': size
                        })
```

```
            return directory_sizes

        except Exception as e:
            return [{'error': str(e)}]

    def check_filesystem_health(self, device):
        """Check filesystem health using fsck"""
        try:
            # Run read-only check
            result = subprocess.run(['fsck', '-n', device],
                                    capture_output=True, text=True)

            return {
                'device': device,
                'status': 'clean' if result.returncode == 0 else 'errors_found',
                'output': result.stdout,
                'errors': result.stderr
            }
        except Exception as e:
            return {'device': device, 'error': str(e)}

    def monitor_filesystem_performance(self, duration=10):
        """Monitor filesystem I/O performance"""
        try:
            # Start iostat monitoring
            process = subprocess.Popen(['iostat', '-x', '1', str(duration)],
                                       stdout=subprocess.PIPE,
                                       stderr=subprocess.PIPE,
                                       text=True)

            output, error = process.communicate()

            return {
                'duration': duration,
                'iostat_output': output,
                'error': error if error else None
            }
        except Exception as e:
            return {'error': str(e)}

    def generate_report(self):
        """Generate comprehensive filesystem report"""
        report = {
            'timestamp': time.strftime('%Y-%m-%d %H:%M:%S'),
            'hostname': os.uname().nodename,
            'mountpoints': self.mountpoints,
            'filesystem_analysis': [],
            'large_files': [],
            'directory_sizes': []
        }

        # Analyze each mounted filesystem
        for mount in self.mountpoints:
            analysis = self.analyze_filesystem(mount['mountpoint'])
            analysis['device'] = mount['device']
            analysis['filesystem_type'] = mount['filesystem']
            report['filesystem_analysis'].append(analysis)

        # Find large files in home and var directories
        for path in ['/home', '/var']:
            if os.path.exists(path):
                large_files = self.get_largest_files(path, min_size_mb=50, count=5)
                report['large_files'].extend(large_files)

        # Get directory sizes for common directories
```

```
        for path in ['/', '/home', '/var', '/usr']:
            if os.path.exists(path):
                dir_sizes = self.get_directory_sizes(path, max_depth=2)
                report['directory_sizes'].extend(dir_sizes)

        return report

# Example usage
if __name__ == "__main__":
    analyzer = FilesystemAnalyzer()

    # Generate and display report
    report = analyzer.generate_report()
    print(json.dumps(report, indent=2))

    # Save report to file
    with open(f'/tmp/filesystem_report_{int(time.time())}.json', 'w') as f:
        json.dump(report, f, indent=2)
```

Bash Filesystem Management Script

```
#!/bin/bash
# filesystem_manager.sh - Comprehensive filesystem management for Ubuntu 22.04

# Configuration
BACKUP_DIR="/backup"
LOG_FILE="/var/log/filesystem_manager.log"

# Colors
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m'

# Logging function
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

# Check if running as root
check_root() {
    if [[ $EUID -ne 0 ]]; then
        echo -e "${RED}This script must be run as root${NC}"
        exit 1
    fi
}

# Display filesystem information
show_filesystem_info() {
    echo -e "${BLUE}=== Filesystem Information ===${NC}"

    echo "Mounted Filesystems:"
    df -hT | grep -v tmpfs
    echo ""

    echo "Filesystem Types:"
    lsblk -f
    echo ""

    echo "Mount Points:"
    mount | grep "^/dev" | column -t
    echo ""
}
```

```
# Create filesystem with optimal settings
create_filesystem() {
    local device="$1"
    local fstype="$2"
    local label="$3"

    if [ -z "$device" ] || [ -z "$fstype" ]; then
        echo "Usage: create_filesystem <device> <filesystem_type> [label]"
        echo "Supported types: ext4, xfs, btrfs"
        return 1
    fi

    echo -e "${YELLOW}Creating $fstype filesystem on $device${NC}"

    case "$fstype" in
        "ext4")
            if [ -n "$label" ]; then
                mkfs.ext4 -L "$label" -m 1 "$device"
            else
                mkfs.ext4 -m 1 "$device"
            fi

            # Optimize for SSD if detected
            if [[ $(cat /sys/block/$(basename $device | sed 's/[0-
9]*$//')/queue/rotational) == "0" ]]; then
                tune2fs -o discard "$device"
                echo "SSD optimizations applied"
            fi
            ;;

        "xfs")
            if [ -n "$label" ]; then
                mkfs.xfs -f -L "$label" "$device"
            else
                mkfs.xfs -f "$device"
            fi
            ;;

        "btrfs")
            if [ -n "$label" ]; then
                mkfs.btrfs -f -L "$label" "$device"
            else
                mkfs.btrfs -f "$device"
            fi
            ;;

        *)
            echo -e "${RED}Unsupported filesystem type: $fstype${NC}"
            return 1
            ;;
    esac

    log_message "Created $fstype filesystem on $device"
    echo -e "${GREEN}Filesystem created successfully${NC}"
}

# Mount filesystem with optimal options
mount_filesystem() {
    local device="$1"
    local mountpoint="$2"
    local fstype="$3"

    if [ -z "$device" ] || [ -z "$mountpoint" ]; then
        echo "Usage: mount_filesystem <device> <mountpoint> [filesystem_type]"
```

```
        return 1
    fi

    # Create mountpoint if it doesn't exist
    mkdir -p "$mountpoint"

    # Detect filesystem type if not provided
    if [ -z "$fstype" ]; then
        fstype=$(blkid -o value -s TYPE "$device")
    fi

    # Set optimal mount options based on filesystem type
    case "$fstype" in
        "ext4")
            mount_options="defaults,noatime"
            # Add discard for SSDs
            if [[ $(cat /sys/block/$(basename $device | sed 's/[0-
9]*$//')/queue/rotational) == "0" ]]; then
                mount_options="$mount_options,discard"
            fi
            ;;
        "xfs")
            mount_options="defaults,noatime"
            ;;
        "btrfs")
            mount_options="defaults,noatime,compress=zstd"
            ;;
        *)
            mount_options="defaults"
            ;;
    esac

    echo -e "${YELLOW}Mounting $device at $mountpoint with options:
$mount_options${NC}"

    if mount -o "$mount_options" "$device" "$mountpoint"; then
        log_message "Mounted $device at $mountpoint"
        echo -e "${GREEN}Filesystem mounted successfully${NC}"

        # Add to fstab for permanent mounting
        uuid=$(blkid -o value -s UUID "$device")
        if [ -n "$uuid" ]; then
            echo "UUID=$uuid $mountpoint $fstype $mount_options 0 2" >> /etc/fstab
            echo "Added to /etc/fstab for automatic mounting"
        fi
    else
        echo -e "${RED}Failed to mount filesystem${NC}"
        return 1
    fi
}

# Check filesystem health
check_filesystem_health() {
    local device="$1"

    if [ -z "$device" ]; then
        echo "Usage: check_filesystem_health <device>"
        return 1
    fi

    echo -e "${BLUE}=== Checking filesystem health for $device ===${NC}"

    # Get filesystem type
    fstype=$(blkid -o value -s TYPE "$device")
```

```
    case "$fstype" in
        "ext4"|"ext3"|"ext2")
            echo "Running fsck for ext filesystem..."
            fsck.ext4 -n "$device"
            ;;
        "xfs")
            echo "Running xfs_check..."
            xfs_check "$device"
            ;;
        "btrfs")
            echo "Running btrfs check..."
            btrfs check "$device"
            ;;
        *)
            echo "Running generic fsck..."
            fsck -n "$device"
            ;;
    esac

    log_message "Filesystem health check completed for $device"
}

# Optimize filesystem performance
optimize_filesystem() {
    local mountpoint="$1"

    if [ -z "$mountpoint" ]; then
        echo "Usage: optimize_filesystem <mountpoint>"
        return 1
    fi

    echo -e "${BLUE}=== Optimizing filesystem at $mountpoint ===${NC}"

    # Get device and filesystem type
    device=$(df "$mountpoint" | tail -1 | awk '{print $1}')
    fstype=$(df -T "$mountpoint" | tail -1 | awk '{print $2}')

    case "$fstype" in
        "ext4")
            echo "Optimizing ext4 filesystem..."

            # Defragment if needed
            e4defrag -c "$mountpoint"

            # Optimize inode allocation
            tune2fs -o journal_data_writeback "$device"

            # Enable TRIM for SSDs
            if [[ $(cat /sys/block/$(basename $device | sed 's/[0-
9]*$//')/queue/rotational) == "0" ]]; then
                tune2fs -o discard "$device"
                echo "SSD optimizations applied"
            fi
            ;;

        "xfs")
            echo "Optimizing XFS filesystem..."

            # XFS is generally well-optimized by default
            # Check if online defragmentation is needed
            xfs_fsr -v "$mountpoint"
            ;;

        "btrfs")
            echo "Optimizing Btrfs filesystem..."
```

```
            # Balance filesystem
            btrfs balance start "$mountpoint"

            # Defragment
            btrfs filesystem defragment -r "$mountpoint"
            ;;
    esac

    log_message "Filesystem optimization completed for $mountpoint"
    echo -e "${GREEN}Optimization completed${NC}"
}

# Backup filesystem
backup_filesystem() {
    local source="$1"
    local backup_name="$2"

    if [ -z "$source" ]; then
        echo "Usage: backup_filesystem <source_mountpoint> [backup_name]"
        return 1
    fi

    if [ -z "$backup_name" ]; then
        backup_name="filesystem_backup_$(date +%Y%m%d_%H%M%S)"
    fi

    backup_path="$BACKUP_DIR/$backup_name"

    echo -e "${YELLOW}Creating backup of $source to $backup_path${NC}"

    # Create backup directory
    mkdir -p "$backup_path"

    # Use rsync for efficient backup
    if rsync -avH --progress "$source/" "$backup_path/"; then
        log_message "Backup created: $source -> $backup_path"
        echo -e "${GREEN}Backup completed successfully${NC}"

        # Create metadata file
        cat > "$backup_path/.backup_info" << EOF
Source: $source
Backup Date: $(date)
Backup Size: $(du -sh "$backup_path" | cut -f1)
EOF
    else
        echo -e "${RED}Backup failed${NC}"
        return 1
    fi
}

# Monitor filesystem usage
monitor_filesystem() {
    echo -e "${BLUE}=== Filesystem Usage Monitor ===${NC}"

    while true; do
        clear
        echo "Filesystem Usage (updated every 5 seconds) - Press Ctrl+C to exit"
        echo "================================================================"

        df -h | grep -E "^Filesystem|^/dev"
        echo ""

        echo "I/O Statistics:"
        iostat -x 1 1 | tail -n +4
```

```
        echo ""

        echo "Top 5 largest directories in /:"
        du -h --max-depth=1 / 2>/dev/null | sort -hr | head -5

        sleep 5
    done
}

# Main menu
case "${1:-help}" in
    "info")
        show_filesystem_info
        ;;
    "create")
        check_root
        create_filesystem "$2" "$3" "$4"
        ;;
    "mount")
        check_root
        mount_filesystem "$2" "$3" "$4"
        ;;
    "check")
        check_root
        check_filesystem_health "$2"
        ;;
    "optimize")
        check_root
        optimize_filesystem "$2"
        ;;
    "backup")
        check_root
        backup_filesystem "$2" "$3"
        ;;
    "monitor")
        monitor_filesystem
        ;;
    "help"|*)
        echo "Filesystem Manager for Ubuntu 22.04"
        echo "Usage: $0 [command] [options]"
        echo ""
        echo "Commands:"
        echo "  info                          - Show filesystem information"
        echo "  create <device> <type> [label] - Create filesystem"
        echo "  mount <device> <mountpoint>   - Mount filesystem with optimal options"
        echo "  check <device>                - Check filesystem health"
        echo "  optimize <mountpoint>         - Optimize filesystem performance"
        echo "  backup <source> [name]        - Backup filesystem"
        echo "  monitor                       - Monitor filesystem usage"
        echo ""
        echo "Supported filesystem types: ext4, xfs, btrfs"
        echo ""
        echo "Examples:"
        echo "  $0 create /dev/sdb1 ext4 MyData"
        echo "  $0 mount /dev/sdb1 /mnt/data"
        echo "  $0 check /dev/sdb1"
        echo "  $0 optimize /home"
        echo "  $0 backup /home home_backup"
        ;;
esac
```

Best Practices for File Systems

Choosing the Right Filesystem

Decision Matrix:

```
Use Case                    | Recommended Filesystem
----------------------------|----------------------
General desktop/server      | ext4
Large files/databases       | XFS
Snapshots/modern features   | Btrfs
Maximum data integrity      | ZFS
Cross-platform compatibility| FAT32/NTFS
High-performance computing  | XFS or ext4
```

Performance Optimization

1. **Mount Options**:

   ```
   # For general use
   mount -o defaults,noatime /dev/sdb1 /mnt/data

   # For SSDs
   mount -o defaults,noatime,discard /dev/sdb1 /mnt/data

   # For databases
   mount -o defaults,noatime,barrier=0 /dev/sdb1 /mnt/database
   ```

2. **I/O Scheduler**:

   ```
   # For SSDs
   echo none > /sys/block/sda/queue/scheduler

   # For HDDs
   echo mq-deadline > /sys/block/sda/queue/scheduler
   ```

3. **Filesystem Tuning**:

   ```
   # Optimize ext4 for SSDs
   tune2fs -o discard /dev/sda1

   # Reduce reserved space (ext4)
   tune2fs -m 1 /dev/sda1
   ```

Security and Maintenance

1. **Regular Checks**:

   ```
   # Schedule regular filesystem checks
   echo "0 3 * * 0 /sbin/fsck -A -f" >> /etc/crontab
   ```

2. **Backup Strategy**:

   ```
   # Automated backups
   rsync -avH /home/ /backup/home/

   # Filesystem snapshots (Btrfs)
   btrfs subvolume snapshot /home /home/.snapshots/$(date +%Y%m%d)
   ```

3. **Monitoring**:

   ```
   # Set up disk usage alerts
   echo "df -h | awk 'NR>1 {if(\$5+0 > 85) print \$0}' | mail -s 'Disk Alert'
   admin@example.com" | crontab -
   ```

Data Organization

Understanding Data Organization

Data organization refers to the systematic arrangement and structuring of data to optimize storage efficiency, access speed, and management ease. Proper data organization is crucial for system performance and data integrity.

Data Organization Fundamentals

What is Data Organization?

Data organization encompasses:

- **Data Structure**: How data is arranged and stored
- **Data Classification**: Categorizing data by type, importance, and access patterns
- **Data Hierarchy**: Organizing data in logical levels and relationships
- **Data Access Patterns**: Understanding how data is accessed and used
- **Data Lifecycle**: Managing data from creation to deletion

Data Classification Types

By Access Frequency

1. **Hot Data**: Frequently accessed, requires fast storage (SSD)
2. **Warm Data**: Occasionally accessed, standard performance storage
3. **Cold Data**: Rarely accessed, slower but cost-effective storage
4. **Archival Data**: Long-term retention, very infrequent access

By Data Type

1. **Structured Data**: Databases, spreadsheets, organized formats
2. **Semi-structured Data**: JSON, XML, log files
3. **Unstructured Data**: Documents, images, videos, emails

By Criticality

1. **Critical Data**: Business-essential, requires high availability
2. **Important Data**: Significant but not critical
3. **Standard Data**: Regular business data
4. **Temporary Data**: Short-term, disposable data

Ubuntu 22.04 Data Organization Tools

Directory Structure Best Practices

```
# Standard Ubuntu directory structure
/
├── home/           # User data
├── var/            # Variable data (logs, databases)
├── usr/            # User programs and libraries
├── opt/            # Optional software packages
├── srv/            # Service data
├── tmp/            # Temporary files
└── mnt/            # Mount points for external storage

# Recommended user data organization
/home/username/
├── Documents/      # Personal documents
├── Projects/       # Development projects
├── Media/          # Photos, videos, music
├── Archive/        # Old/archived files
├── Backup/         # Local backups
└── Work/           # Work-related files
```

File Organization Commands

```
# Create organized directory structure
mkdir -p ~/Documents/{Personal,Work}/{2024,2023,Archive}
mkdir -p ~/Projects/{Active,Completed,Archive}
mkdir -p ~/Media/{Photos,Videos,Music}/{2024,2023,Archive}

# File organization by date
find ~/Downloads -type f -newermt "2024-01-01" ! -newermt "2024-12-31" \
    -exec mkdir -p ~/Archive/2024/ \; -exec mv {} ~/Archive/2024/ \;

# Organize files by extension
cd ~/Downloads
for ext in pdf doc docx txt; do
    mkdir -p ~/Documents/$(echo $ext | tr '[:lower:]' '[:upper:]')
    find . -name "*.$ext" -exec mv {} ~/Documents/$(echo $ext | tr '[:lower:]'
'[:upper:]')/ \;
done

# Find and organize duplicate files
fdupes -r ~/Documents -d
```

Data Backup and Archival

Backup Strategies

```
# 3-2-1 Backup Rule Implementation
# 3 copies, 2 different media types, 1 offsite

# Local backup (rsync)
rsync -avH --delete ~/Documents/ /backup/documents/

# External drive backup
rsync -avH --delete ~/Documents/ /media/external/backup/documents/

# Cloud backup (rclone)
rclone sync ~/Documents/ cloud:backup/documents/

# Compressed archive backup
tar -czf /backup/documents_$(date +%Y%m%d).tar.gz ~/Documents/
```

Automated Data Organization

```
# Create organization script
cat > ~/bin/organize_data.sh << 'EOF'
#!/bin/bash

# Organize downloads by file type
organize_downloads() {
    cd ~/Downloads

    # Create directories
    mkdir -p {Documents,Images,Videos,Music,Archives,Software}

    # Move files by type
    find . -maxdepth 1 -type f \( -iname "*.pdf" -o -iname "*.doc" -o -iname "*.docx"
-o -iname "*.txt" \) -exec mv {} Documents/ \;
    find . -maxdepth 1 -type f \( -iname "*.jpg" -o -iname "*.png" -o -iname "*.gif" -
o -iname "*.jpeg" \) -exec mv {} Images/ \;
    find . -maxdepth 1 -type f \( -iname "*.mp4" -o -iname "*.avi" -o -iname "*.mkv"
\) -exec mv {} Videos/ \;
    find . -maxdepth 1 -type f \( -iname "*.mp3" -o -iname "*.flac" -o -iname "*.wav"
\) -exec mv {} Music/ \;
    find . -maxdepth 1 -type f \( -iname "*.zip" -o -iname "*.tar.gz" -o -iname
"*.rar" \) -exec mv {} Archives/ \;
    find . -maxdepth 1 -type f \( -iname "*.deb" -o -iname "*.appimage" \) -exec mv {}
Software/ \;
}

# Clean old temporary files
clean_temp() {
    find ~/Downloads -type f -mtime +30 -delete
    find ~/.cache -type f -mtime +7 -delete
    find /tmp -user $USER -type f -mtime +1 -delete 2>/dev/null
}

# Archive old files
archive_old() {
    find ~/Documents -type f -mtime +365 -exec mkdir -p ~/Archive/$(date +%Y) \; -exec
mv {} ~/Archive/$(date +%Y)/ \;
}

# Execute functions
organize_downloads
clean_temp
archive_old

echo "Data organization completed: $(date)"
EOF

chmod +x ~/bin/organize_data.sh

# Schedule with cron
echo "0 2 * * * ~/bin/organize_data.sh" | crontab -
```

Data Compression and Deduplication

File Compression Techniques

```
# Install compression tools
sudo apt install p7zip-full rar unrar gzip bzip2 xz-utils

# Create compressed archives
# gzip (fast, good compression)
tar -czf archive.tar.gz /path/to/data

# bzip2 (better compression, slower)
```

```
tar -cjf archive.tar.bz2 /path/to/data

# xz (best compression, slowest)
tar -cJf archive.tar.xz /path/to/data

# 7zip (excellent compression)
7z a -t7z -mx=9 archive.7z /path/to/data

# Compare compression ratios
for method in gz bz2 xz 7z; do
    echo "Compressing with $method..."
    case $method in
        gz) tar -czf test.tar.gz ~/Documents ;;
        bz2) tar -cjf test.tar.bz2 ~/Documents ;;
        xz) tar -cJf test.tar.xz ~/Documents ;;
        7z) 7z a test.7z ~/Documents ;;
    esac
    echo "$method: $(ls -lh test.*$method* | awk '{print $5}')"
done
```

Duplicate File Management

```
# Install deduplication tools
sudo apt install fdupes rdfind

# Find duplicates with fdupes
fdupes -r ~/Documents

# Find and delete duplicates interactively
fdupes -r -d ~/Documents

# Find duplicates with rdfind
rdfind ~/Documents

# Advanced duplicate handling with rdfind
rdfind -makehardlinks true ~/Documents
rdfind -makeresultsfile false -makehardlinks true ~/Documents
```

Data Monitoring and Analysis

Storage Usage Analysis

```
# Analyze disk usage by directory
ncdu /home

# Find largest files
find /home -type f -exec du -h {} + | sort -rh | head -20

# Analyze file types and sizes
find /home -type f | sed 's/.*\\.//' | sort | uniq -c | sort -rn

# Monitor storage usage trends
cat > ~/bin/storage_trend.sh << 'EOF'
#!/bin/bash

LOGFILE="/var/log/storage_usage.log"

echo "$(date): $(df -h / | tail -1 | awk '{print $5}')" >> $LOGFILE

# Generate weekly report
if [ "$(date +%u)" -eq 1 ]; then
    echo "Weekly Storage Report - $(date)" > /tmp/storage_report.txt
    tail -7 $LOGFILE >> /tmp/storage_report.txt
```

```
    mail -s "Storage Usage Report" admin@example.com < /tmp/storage_report.txt
fi
EOF

chmod +x ~/bin/storage_trend.sh
echo "0 0 * * * ~/bin/storage_trend.sh" | crontab -
```

File System Metadata Analysis

```
# Analyze inode usage
df -i

# Find directories with most files
find /home -type d -exec sh -c 'echo "$(find "$1" -maxdepth 1 | wc -l) $1"' _ {} \; |
sort -rn | head -10

# Analyze file access patterns
find /home -type f -atime -7 | wc -l  # Files accessed in last 7 days
find /home -type f -mtime -7 | wc -l  # Files modified in last 7 days
find /home -type f -atime +365 | wc -l  # Files not accessed in over a year
```

Frequently Asked Questions

Q: How should I organize my home directory in Ubuntu 22.04?

A: Follow this recommended structure:

```
~/
├── Documents/
│   ├── Personal/
│   ├── Work/
│   └── Archive/
├── Projects/
│   ├── Active/
│   ├── Completed/
│   └── Learning/
├── Media/
│   ├── Photos/
│   ├── Videos/
│   └── Music/
├── Downloads/
│   └── (temporary, clean regularly)
├── Backup/
│   └── (local backup copies)
└── Scripts/
    └── (personal automation scripts)
```

Q: What's the best way to handle duplicate files?

A: Use these strategies:

```
# 1. Find duplicates without deleting
fdupes -r ~/Documents

# 2. Interactive deletion
fdupes -r -d ~/Documents

# 3. Automatic deletion (keep first occurrence)
fdupes -r -f ~/Documents | grep -v '^$' | xargs rm

# 4. Convert to hard links (saves space)
rdfind -makehardlinks true ~/Documents
```

Q: How can I automate file organization?

A: Create automated organization scripts:

```
# File organizer by extension
#!/bin/bash
organize_by_extension() {
    local source_dir="$1"
    local target_dir="$2"

    cd "$source_dir"

    for file in *.*; do
        if [ -f "$file" ]; then
            ext="${file##*.}"
            ext_upper=$(echo "$ext" | tr '[:lower:]' '[:upper:]')
            mkdir -p "$target_dir/$ext_upper"
            mv "$file" "$target_dir/$ext_upper/"
        fi
    done
}

# Usage
organize_by_extension ~/Downloads ~/Downloads/Organized
```

Q: How do I implement a data retention policy?

A: Create a retention management system:

```
#!/bin/bash
# data_retention.sh

# Define retention periods (days)
TEMP_RETENTION=7
LOG_RETENTION=90
BACKUP_RETENTION=365
ARCHIVE_RETENTION=2555  # 7 years

# Clean temporary files
find /tmp -type f -mtime +$TEMP_RETENTION -delete
find ~/.cache -type f -mtime +$TEMP_RETENTION -delete

# Archive old logs
find /var/log -name "*.log" -mtime +$LOG_RETENTION -gzip

# Remove old backups
find /backup -name "*.tar.gz" -mtime +$BACKUP_RETENTION -delete

# Alert for very old archives
find /archive -type f -mtime +$ARCHIVE_RETENTION -ls | \
    mail -s "Files exceeding retention policy" admin@example.com
```

Coding Examples

Python Data Organization Manager

```
#!/usr/bin/env python3
"""
Advanced data organization manager for Ubuntu 22.04
"""
import os
import shutil
import hashlib
```

```
import json
import time
import mimetypes
from pathlib import Path
from collections import defaultdict
import argparse

class DataOrganizer:
    def __init__(self, base_path=None):
        self.base_path = Path(base_path) if base_path else Path.home()
        self.file_types = {
            'documents': ['.pdf', '.doc', '.docx', '.txt', '.rtf', '.odt'],
            'images': ['.jpg', '.jpeg', '.png', '.gif', '.bmp', '.tiff', '.svg'],
            'videos': ['.mp4', '.avi', '.mkv', '.mov', '.wmv', '.flv', '.webm'],
            'music': ['.mp3', '.flac', '.wav', '.ogg', '.m4a', '.aac'],
            'archives': ['.zip', '.tar', '.gz', '.bz2', '.xz', '.rar', '.7z'],
            'code': ['.py', '.js', '.html', '.css', '.java', '.cpp', '.c', '.php'],
            'data': ['.csv', '.json', '.xml', '.sql', '.db', '.sqlite']
        }
        self.stats = defaultdict(int)

    def get_file_hash(self, filepath):
        """Calculate MD5 hash of a file"""
        hash_md5 = hashlib.md5()
        try:
            with open(filepath, "rb") as f:
                for chunk in iter(lambda: f.read(4096), b""):
                    hash_md5.update(chunk)
            return hash_md5.hexdigest()
        except (IOError, OSError):
            return None

    def find_duplicates(self, directory):
        """Find duplicate files in a directory"""
        duplicates = defaultdict(list)

        for root, dirs, files in os.walk(directory):
            for file in files:
                filepath = Path(root) / file
                if filepath.is_file():
                    file_hash = self.get_file_hash(filepath)
                    if file_hash:
                        duplicates[file_hash].append(filepath)

        # Filter out unique files
        return {k: v for k, v in duplicates.items() if len(v) > 1}

    def organize_by_type(self, source_dir, target_dir=None):
        """Organize files by type"""
        source_path = Path(source_dir)
        target_path = Path(target_dir) if target_dir else source_path / "Organized"

        if not source_path.exists():
            raise ValueError(f"Source directory {source_path} does not exist")

        target_path.mkdir(exist_ok=True)

        for file_path in source_path.rglob('*'):
            if file_path.is_file() and file_path.parent == source_path:
                file_ext = file_path.suffix.lower()

                # Determine file category
                category = 'misc'
                for cat, extensions in self.file_types.items():
                    if file_ext in extensions:
```

```
                        category = cat
                        break

                # Create category directory
                category_dir = target_path / category.title()
                category_dir.mkdir(exist_ok=True)

                # Move file
                new_path = category_dir / file_path.name

                # Handle naming conflicts
                counter = 1
                original_name = new_path.stem
                while new_path.exists():
                    new_path = category_dir / f"{original_name}_{counter}
{file_path.suffix}"
                    counter += 1

                try:
                    shutil.move(str(file_path), str(new_path))
                    self.stats[f'moved_{category}'] += 1
                    print(f"Moved {file_path.name} to {category.title()}/")
                except (IOError, OSError) as e:
                    print(f"Error moving {file_path}: {e}")

    def organize_by_date(self, source_dir, target_dir=None):
        """Organize files by modification date"""
        source_path = Path(source_dir)
        target_path = Path(target_dir) if target_dir else source_path / "ByDate"

        target_path.mkdir(exist_ok=True)

        for file_path in source_path.rglob('*'):
            if file_path.is_file():
                # Get modification time
                mtime = file_path.stat().st_mtime
                date_str = time.strftime('%Y/%m', time.localtime(mtime))

                # Create date directory
                date_dir = target_path / date_str
                date_dir.mkdir(parents=True, exist_ok=True)

                # Move file
                new_path = date_dir / file_path.name

                # Handle naming conflicts
                counter = 1
                original_name = new_path.stem
                while new_path.exists():
                    new_path = date_dir / f"{original_name}_{counter}
{file_path.suffix}"
                    counter += 1

                try:
                    shutil.move(str(file_path), str(new_path))
                    self.stats['moved_by_date'] += 1
                except (IOError, OSError) as e:
                    print(f"Error moving {file_path}: {e}")

    def clean_empty_directories(self, directory):
        """Remove empty directories"""
        removed_count = 0
        for root, dirs, files in os.walk(directory, topdown=False):
            for dir_name in dirs:
                dir_path = Path(root) / dir_name
```

```
                try:
                    if not any(dir_path.iterdir()):
                        dir_path.rmdir()
                        removed_count += 1
                        print(f"Removed empty directory: {dir_path}")
                except OSError:
                    pass  # Directory not empty or permission denied

        self.stats['removed_empty_dirs'] = removed_count

    def analyze_directory(self, directory):
        """Analyze directory structure and file distribution"""
        analysis = {
            'total_files': 0,
            'total_size': 0,
            'file_types': defaultdict(int),
            'size_by_type': defaultdict(int),
            'largest_files': [],
            'oldest_files': [],
            'newest_files': []
        }

        files_with_info = []

        for root, dirs, files in os.walk(directory):
            for file in files:
                file_path = Path(root) / file
                if file_path.is_file():
                    try:
                        stat_info = file_path.stat()
                        file_size = stat_info.st_size
                        file_ext = file_path.suffix.lower()

                        analysis['total_files'] += 1
                        analysis['total_size'] += file_size
                        analysis['file_types'][file_ext] += 1
                        analysis['size_by_type'][file_ext] += file_size

                        files_with_info.append({
                            'path': file_path,
                            'size': file_size,
                            'mtime': stat_info.st_mtime
                        })
                    except OSError:
                        continue

        # Sort files for top lists
        files_with_info.sort(key=lambda x: x['size'], reverse=True)
        analysis['largest_files'] = [
            {'path': str(f['path']), 'size_mb': f['size'] / (1024*1024)}
            for f in files_with_info[:10]
        ]

        files_with_info.sort(key=lambda x: x['mtime'])
        analysis['oldest_files'] = [
            {'path': str(f['path']), 'date': time.strftime('%Y-%m-%d',
time.localtime(f['mtime']))}
            for f in files_with_info[:10]
        ]

        files_with_info.sort(key=lambda x: x['mtime'], reverse=True)
        analysis['newest_files'] = [
            {'path': str(f['path']), 'date': time.strftime('%Y-%m-%d',
time.localtime(f['mtime']))}
            for f in files_with_info[:10]
```

```
        ]

        return analysis

    def create_backup_structure(self, source_dir, backup_dir):
        """Create organized backup structure"""
        source_path = Path(source_dir)
        backup_path = Path(backup_dir)

        timestamp = time.strftime('%Y%m%d_%H%M%S')
        backup_target = backup_path / f"backup_{timestamp}"
        backup_target.mkdir(parents=True, exist_ok=True)

        # Copy files with organization
        for category, extensions in self.file_types.items():
            category_backup = backup_target / category
            category_backup.mkdir(exist_ok=True)

            for root, dirs, files in os.walk(source_path):
                for file in files:
                    file_path = Path(root) / file
                    if file_path.suffix.lower() in extensions:
                        try:
                            shutil.copy2(file_path, category_backup)
                            self.stats[f'backed_up_{category}'] += 1
                        except (IOError, OSError) as e:
                            print(f"Error backing up {file_path}: {e}")

    def generate_report(self):
        """Generate organization report"""
        report = {
            'timestamp': time.strftime('%Y-%m-%d %H:%M:%S'),
            'statistics': dict(self.stats),
            'summary': f"Processed {sum(self.stats.values())} operations"
        }

        return json.dumps(report, indent=2)

def main():
    parser = argparse.ArgumentParser(description='Data Organization Manager')
    parser.add_argument('command', choices=['organize', 'analyze', 'duplicates',
'backup', 'clean'])
    parser.add_argument('source', help='Source directory')
    parser.add_argument('--target', help='Target directory')
    parser.add_argument('--by-type', action='store_true', help='Organize by file
type')
    parser.add_argument('--by-date', action='store_true', help='Organize by date')

    args = parser.parse_args()

    organizer = DataOrganizer()

    if args.command == 'organize':
        if args.by_type:
            organizer.organize_by_type(args.source, args.target)
        elif args.by_date:
            organizer.organize_by_date(args.source, args.target)
        else:
            organizer.organize_by_type(args.source, args.target)

    elif args.command == 'analyze':
        analysis = organizer.analyze_directory(args.source)
        print(json.dumps(analysis, indent=2, default=str))

    elif args.command == 'duplicates':
```

```
        duplicates = organizer.find_duplicates(args.source)
        print(f"Found {len(duplicates)} sets of duplicate files:")
        for hash_val, files in duplicates.items():
            print(f"Hash {hash_val[:8]}...")
            for file in files:
                print(f"  {file}")

    elif args.command == 'backup':
        if not args.target:
            print("Target directory required for backup")
            return
        organizer.create_backup_structure(args.source, args.target)

    elif args.command == 'clean':
        organizer.clean_empty_directories(args.source)

    print(organizer.generate_report())

if __name__ == "__main__":
    main()
```

Bash File Organization System

```
#!/bin/bash
# advanced_file_organizer.sh - Advanced file organization system for Ubuntu 22.04

# Configuration
CONFIG_FILE="$HOME/.file_organizer.conf"
LOG_FILE="$HOME/.file_organizer.log"

# Default configuration
cat > "$CONFIG_FILE" << 'EOF'
# File Organization Configuration
ORGANIZE_DOWNLOADS=true
ORGANIZE_DESKTOP=true
AUTO_ARCHIVE_DAYS=365
DUPLICATE_ACTION=ask  # ask, delete, link
BACKUP_ENABLED=true
BACKUP_DIR="$HOME/Backup"

# File type definitions
DOCUMENTS="pdf doc docx txt rtf odt"
IMAGES="jpg jpeg png gif bmp tiff svg"
VIDEOS="mp4 avi mkv mov wmv flv webm"
MUSIC="mp3 flac wav ogg m4a aac"
ARCHIVES="zip tar gz bz2 xz rar 7z"
CODE="py js html css java cpp c php"
DATA="csv json xml sql db sqlite"
EOF

# Load configuration
source "$CONFIG_FILE"

# Logging function
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" >> "$LOG_FILE"
    echo "$1"
}

# Create directory structure
create_structure() {
    local base_dir="$1"

    log_message "Creating directory structure in $base_dir"
```

```
    mkdir -p "$base_dir"/{Documents,Images,Videos,Music,Archives,Code,Data,Misc}
    mkdir -p "$base_dir"/Archive/{$(date +%Y),$(date -d 'last year' +%Y)}
    mkdir -p "$base_dir"/Backup

    log_message "Directory structure created"
}

# Get file category
get_file_category() {
    local file="$1"
    local extension="${file##*.}"
    extension=$(echo "$extension" | tr '[:upper:]' '[:lower:]')

    case " $DOCUMENTS " in *" $extension "*) echo "Documents"; return ;; esac
    case " $IMAGES " in *" $extension "*) echo "Images"; return ;; esac
    case " $VIDEOS " in *" $extension "*) echo "Videos"; return ;; esac
    case " $MUSIC " in *" $extension "*) echo "Music"; return ;; esac
    case " $ARCHIVES " in *" $extension "*) echo "Archives"; return ;; esac
    case " $CODE " in *" $extension "*) echo "Code"; return ;; esac
    case " $DATA " in *" $extension "*) echo "Data"; return ;; esac

    echo "Misc"
}

# Organize files by type
organize_by_type() {
    local source_dir="$1"
    local target_dir="$2"

    log_message "Organizing files from $source_dir to $target_dir"

    create_structure "$target_dir"

    local moved_count=0

    find "$source_dir" -maxdepth 1 -type f | while read -r file; do
        if [[ -f "$file" ]]; then
            local filename=$(basename "$file")
            local category=$(get_file_category "$filename")
            local target_file="$target_dir/$category/$filename"

            # Handle name conflicts
            local counter=1
            local base_name="${filename%.*}"
            local extension="${filename##*.}"

            while [[ -e "$target_file" ]]; do
                if [[ "$extension" != "$filename" ]]; then

target_file="$target_dir/$category/${base_name}_${counter}.${extension}"
                else
                    target_file="$target_dir/$category/${filename}_${counter}"
                fi
                ((counter++))
            done

            if mv "$file" "$target_file"; then
                log_message "Moved $filename to $category/"
                ((moved_count++))
            else
                log_message "Failed to move $filename"
            fi
        fi
    done
```

```
    log_message "Organized $moved_count files"
}

# Find and handle duplicates
handle_duplicates() {
    local directory="$1"
    local action="${2:-$DUPLICATE_ACTION}"

    log_message "Finding duplicates in $directory"

    if ! command -v fdupes &> /dev/null; then
        log_message "Installing fdupes for duplicate detection"
        sudo apt update && sudo apt install -y fdupes
    fi

    local duplicates_file="/tmp/duplicates_$$"
    fdupes -r "$directory" > "$duplicates_file"

    local duplicate_sets=0
    local files_processed=0

    while IFS= read -r line; do
        if [[ -z "$line" ]]; then
            ((duplicate_sets++))
        elif [[ -f "$line" ]]; then
            case "$action" in
                "delete")
                    if [[ $files_processed -gt 0 ]]; then
                        rm "$line"
                        log_message "Deleted duplicate: $line"
                    fi
                    ;;
                "link")
                    if [[ $files_processed -gt 0 ]]; then
                        local first_file=$(head -1 "$duplicates_file")
                        rm "$line"
                        ln "$first_file" "$line"
                        log_message "Linked duplicate: $line"
                    fi
                    ;;
                "ask")
                    if [[ $files_processed -gt 0 ]]; then
                        echo "Duplicate found: $line"
                        read -p "Delete this file? (y/N): " -n 1 -r
                        echo
                        if [[ $REPLY =~ ^[Yy]$ ]]; then
                            rm "$line"
                            log_message "User deleted duplicate: $line"
                        fi
                    fi
                    ;;
            esac
            ((files_processed++))
        fi
    done < "$duplicates_file"

    rm "$duplicates_file"
    log_message "Processed $duplicate_sets sets of duplicates"
}

# Archive old files
archive_old_files() {
    local source_dir="$1"
    local days="${2:-$AUTO_ARCHIVE_DAYS}"
```

```
    log_message "Archiving files older than $days days from $source_dir"

    local archive_dir="$source_dir/Archive/$(date +%Y)"
    mkdir -p "$archive_dir"

    local archived_count=0

    find "$source_dir" -type f -mtime +$days ! -path "*/Archive/*" ! -path
"*/Backup/*" | while read -r file; do
        local relative_path="${file#$source_dir/}"
        local archive_path="$archive_dir/$relative_path"
        local archive_parent=$(dirname "$archive_path")

        mkdir -p "$archive_parent"

        if mv "$file" "$archive_path"; then
            log_message "Archived: $relative_path"
            ((archived_count++))
        fi
    done

    log_message "Archived $archived_count files"
}

# Create backup
create_backup() {
    local source_dir="$1"
    local backup_dir="${2:-$BACKUP_DIR}"

    if [[ "$BACKUP_ENABLED" != "true" ]]; then
        log_message "Backup disabled in configuration"
        return
    fi

    log_message "Creating backup of $source_dir to $backup_dir"

    local timestamp=$(date +%Y%m%d_%H%M%S)
    local backup_target="$backup_dir/backup_$timestamp"

    mkdir -p "$backup_target"

    if rsync -av --progress "$source_dir/" "$backup_target/"; then
        log_message "Backup created successfully: $backup_target"

        # Compress backup
        if command -v tar &> /dev/null; then
            tar -czf "$backup_target.tar.gz" -C "$backup_dir" "backup_$timestamp"
            rm -rf "$backup_target"
            log_message "Backup compressed: $backup_target.tar.gz"
        fi
    else
        log_message "Backup failed"
    fi
}

# Generate organization report
generate_report() {
    local directory="$1"
    local report_file="/tmp/organization_report_$(date +%Y%m%d).txt"

    {
        echo "File Organization Report"
        echo "========================"
        echo "Generated: $(date)"
```

```
        echo "Directory: $directory"
        echo ""

        echo "Directory Structure:"
        tree -d -L 2 "$directory" 2>/dev/null || find "$directory" -type d | head -20
        echo ""

        echo "File Count by Type:"
        for type in Documents Images Videos Music Archives Code Data Misc; do
            if [[ -d "$directory/$type" ]]; then
                count=$(find "$directory/$type" -type f | wc -l)
                echo "$type: $count files"
            fi
        done
        echo ""

        echo "Storage Usage:"
        du -sh "$directory"/* 2>/dev/null | sort -hr
        echo ""

        echo "Recent Activity (from log):"
        tail -20 "$LOG_FILE"

    } > "$report_file"

    echo "Report generated: $report_file"

    # Email report if mail is configured
    if command -v mail &> /dev/null; then
        cat "$report_file" | mail -s "File Organization Report" "$USER@localhost"
    fi
}

# Main execution
main() {
    case "${1:-help}" in
        "organize")
            organize_by_type "${2:-$HOME/Downloads}" "${3:-$HOME/Organized}"
            ;;
        "duplicates")
            handle_duplicates "${2:-$HOME}" "${3:-ask}"
            ;;
        "archive")
            archive_old_files "${2:-$HOME}" "${3:-365}"
            ;;
        "backup")
            create_backup "${2:-$HOME}" "${3:-$BACKUP_DIR}"
            ;;
        "report")
            generate_report "${2:-$HOME}"
            ;;
        "full")
            # Full organization workflow
            local target_dir="${2:-$HOME/Organized}"

            log_message "Starting full organization workflow"

            create_backup "$HOME/Downloads"
            organize_by_type "$HOME/Downloads" "$target_dir"
            handle_duplicates "$target_dir" "ask"
            archive_old_files "$target_dir"
            generate_report "$target_dir"

            log_message "Full organization workflow completed"
            ;;
```

```
        "config")
            nano "$CONFIG_FILE"
            ;;
        "help"|*)
            echo "Advanced File Organizer for Ubuntu 22.04"
            echo "Usage: $0 [command] [options]"
            echo ""
            echo "Commands:"
            echo "  organize [source] [target]    - Organize files by type"
            echo "  duplicates [dir] [action]     - Handle duplicate files
(ask/delete/link)"
            echo "  archive [dir] [days]          - Archive old files"
            echo "  backup [source] [target]      - Create backup"
            echo "  report [directory]            - Generate organization report"
            echo "  full [target]                 - Run complete organization
workflow"
            echo "  config                        - Edit configuration"
            echo ""
            echo "Examples:"
            echo "  $0 organize ~/Downloads ~/Organized"
            echo "  $0 duplicates ~/Documents delete"
            echo "  $0 archive ~/Documents 180"
            echo "  $0 full ~/MyFiles"
            ;;
    esac
}

# Run main function with all arguments
main "$@"
```

Best Practices for Data Organization

Strategic Planning

1. **Data Classification Strategy**:
    - Identify data types and access patterns
    - Define retention policies
    - Establish backup requirements
    - Plan for data growth
2. **Directory Structure Design**:
    - Use consistent naming conventions
    - Implement logical hierarchies
    - Plan for scalability
    - Document organization rules

Automation and Maintenance

1. **Automated Organization**:

```
# Daily organization cron job
0 2 * * * /home/user/bin/organize_data.sh

# Weekly duplicate cleanup
```

```
0 1 * * 0 fdupes -r -d /home/user/Documents

# Monthly archival
0 0 1 * * /home/user/bin/archive_old_files.sh
```

2. **Monitoring and Alerts**:

```
# Storage usage monitoring
if [ $(df / | tail -1 | awk '{print $5}' | sed 's/%//') -gt 85 ]; then
    echo "Disk usage critical" | mail -s "Storage Alert" admin
fi
```

Security and Compliance

1. **Data Protection**:
   - Implement access controls
   - Use encryption for sensitive data
   - Regular security audits
   - Backup verification
2. **Compliance Requirements**:
   - Document data handling procedures
   - Implement retention policies
   - Audit trail maintenance
   - Regular compliance reviews

```
# Set up secure data handling
# Encrypt sensitive directories
sudo apt install ecryptfs-utils
ecryptfs-migrate-home -u username

# Set appropriate permissions
find /home/user/Documents -type f -exec chmod 640 {} \;
find /home/user/Documents -type d -exec chmod 750 {} \;
```

RAID Systems

Understanding RAID

RAID (Redundant Array of Independent Disks) is a technology that combines multiple disk drives into a single logical unit to improve performance, provide redundancy, or both. RAID systems are crucial for data protection and performance optimization in storage systems.

What is RAID?

RAID provides:

- **Data Redundancy**: Protection against disk failures
- **Performance Improvement**: Faster read/write operations through parallelism

- **Storage Efficiency**: Optimized use of available disk space
- **Fault Tolerance**: Ability to continue operation despite disk failures
- **Scalability**: Easy expansion of storage capacity

RAID Levels Overview

Standard RAID Levels

```
RAID Level    | Description              | Min Disks | Fault Tolerance | Performance
--------------|--------------------------|-----------|-----------------|-------------
RAID 0        | Striping (no redundancy) | 2         | None            | High
Read/Write
RAID 1        | Mirroring                | 2         | 1 disk failure | Good Read,
Normal Write
RAID 5        | Striping with Parity     | 3         | 1 disk failure | Good Read,
Moderate Write
RAID 6        | Double Parity            | 4         | 2 disk failures| Good Read,
Lower Write
RAID 10       | Mirror + Stripe          | 4         | Multiple disks | High Read/Write
```

RAID 0 - Striping

Characteristics: * Data is striped across multiple disks * No redundancy - failure of any disk results in total data loss * Excellent performance for both reads and writes * Full utilization of disk capacity

Use Cases: * High-performance applications * Temporary data processing * Non-critical data requiring high speed

```
# Create RAID 0 with mdadm
sudo mdadm --create /dev/md0 --level=0 --raid-devices=2 /dev/sdb /dev/sdc

# Format and mount
sudo mkfs.ext4 /dev/md0
sudo mkdir /mnt/raid0
sudo mount /dev/md0 /mnt/raid0
```

RAID 1 - Mirroring

Characteristics: * Data is mirrored across multiple disks * Can survive failure of all but one disk * Read performance can be improved * Write performance is similar to single disk * 50% storage efficiency

Use Cases: * Critical data requiring high availability * Operating system partitions * Database storage * Boot drives

```
# Create RAID 1 with mdadm
sudo mdadm --create /dev/md1 --level=1 --raid-devices=2 /dev/sdb /dev/sdc

# Monitor RAID status
cat /proc/mdstat

# Check RAID details
sudo mdadm --detail /dev/md1
```

RAID 5 - Striping with Parity

Characteristics: * Data and parity information striped across all disks * Can

survive failure of one disk * Good read performance * Write performance penalty due to parity calculation * Storage efficiency: (n-1)/n where n is number of disks

Use Cases: * General purpose storage * File servers * Backup storage * Cost-effective redundancy

```
# Create RAID 5 with 3 disks
sudo mdadm --create /dev/md5 --level=5 --raid-devices=3 /dev/sdb /dev/sdc /dev/sdd

# Add spare disk
sudo mdadm --add /dev/md5 /dev/sde
```

RAID 6 - Double Parity

Characteristics: * Two parity blocks for each stripe * Can survive failure of two disks * Better fault tolerance than RAID 5 * Lower write performance than RAID 5 * Storage efficiency: (n-2)/n

Use Cases: * Large disk arrays * Critical data with high availability requirements * Long rebuild times scenarios * Enterprise storage systems

```
# Create RAID 6 with 4 disks
sudo mdadm --create /dev/md6 --level=6 --raid-devices=4 /dev/sdb /dev/sdc /dev/sdd
/dev/sde
```

RAID 10 - Mirror and Stripe

Characteristics: * Combines RAID 1 (mirroring) and RAID 0 (striping) * Excellent performance and redundancy * Can survive multiple disk failures (in different mirror sets) * 50% storage efficiency * Fast rebuild times

Use Cases: * High-performance databases * Virtual machine storage * High-availability applications * Enterprise critical data

```
# Create RAID 10 with 4 disks
sudo mdadm --create /dev/md10 --level=10 --raid-devices=4 /dev/sdb /dev/sdc /dev/sdd
/dev/sde
```

Setting Up RAID on Ubuntu 22.04

Installing RAID Tools

```
# Install mdadm for software RAID
sudo apt update
sudo apt install mdadm

# Install additional tools
sudo apt install smartmontools hdparm parted

# Check available disks
sudo fdisk -l
lsblk
```

Preparing Disks for RAID

```
# Wipe disk signatures (WARNING: This destroys data!)
sudo wipefs -a /dev/sdb
sudo wipefs -a /dev/sdc

# Create partitions (optional, can use whole disks)
```

```
sudo parted /dev/sdb mklabel gpt
sudo parted /dev/sdb mkpart primary 0% 100%
sudo parted /dev/sdb set 1 raid on

# Verify no existing RAID metadata
sudo mdadm --examine /dev/sdb
sudo mdadm --examine /dev/sdc
```

Creating RAID Arrays

```
# Create RAID 1 array
sudo mdadm --create --verbose /dev/md0 \\
    --level=1 \\
    --raid-devices=2 \\
    /dev/sdb /dev/sdc

# Create RAID 5 array with spare
sudo mdadm --create --verbose /dev/md1 \\
    --level=5 \\
    --raid-devices=3 \\
    --spare-devices=1 \\
    /dev/sdb /dev/sdc /dev/sdd /dev/sde

# Save RAID configuration
sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf

# Update initramfs
sudo update-initramfs -u
```

RAID Management and Monitoring

Monitoring RAID Status

```
# Check RAID status
cat /proc/mdstat

# Detailed RAID information
sudo mdadm --detail /dev/md0

# Monitor RAID in real-time
watch cat /proc/mdstat

# Check individual disk health
sudo smartctl -a /dev/sdb
```

RAID Maintenance Operations

```
# Add a disk to RAID array
sudo mdadm --add /dev/md0 /dev/sdd

# Remove a disk from RAID array
sudo mdadm --remove /dev/md0 /dev/sdd

# Mark a disk as failed
sudo mdadm --fail /dev/md0 /dev/sdb

# Replace a failed disk
sudo mdadm --remove /dev/md0 /dev/sdb
# Physically replace disk
sudo mdadm --add /dev/md0 /dev/sdb

# Grow RAID array (add more disks)
```

```
sudo mdadm --grow /dev/md0 --raid-devices=3 --add /dev/sdd

# Reshape RAID array
sudo mdadm --grow /dev/md0 --level=6
```

RAID Performance Optimization

Stripe Size Optimization

```
# Create RAID with custom chunk size
sudo mdadm --create /dev/md0 \\
    --level=5 \\
    --chunk=64 \\
    --raid-devices=3 \\
    /dev/sdb /dev/sdc /dev/sdd

# Test different chunk sizes for your workload
for chunk in 32 64 128 256 512; do
    echo "Testing chunk size: $chunk"
    # Create test array and measure performance
done
```

I/O Scheduler Optimization

```
# Set I/O scheduler for RAID devices
echo mq-deadline | sudo tee /sys/block/md0/queue/scheduler

# Optimize readahead
sudo blockdev --setra 65536 /dev/md0

# Disable barriers for better performance (if UPS protected)
sudo mount -o barrier=0 /dev/md0 /mnt/raid
```

Filesystem Considerations

```
# Align filesystem to RAID geometry
# For RAID 5 with 3 disks and 64K chunk size:
# Stripe width = (disks - parity) * chunk_size = 2 * 64K = 128K

sudo mkfs.ext4 -E stride=16,stripe-width=32 /dev/md0

# For XFS on RAID
sudo mkfs.xfs -d su=65536,sw=2 /dev/md0
```

Frequently Asked Questions

Q: Which RAID level should I choose for my use case?

A: Choose based on your priorities:

```
Priority              | Recommended RAID
----------------------|------------------
Maximum Performance   | RAID 0 (no redundancy)
High Availability     | RAID 1 or RAID 10
Balanced Performance  | RAID 5
Maximum Protection    | RAID 6
Performance + Safety  | RAID 10
```

Q: Can I convert between RAID levels?

A: Yes, but with limitations:

```
# Convert RAID 1 to RAID 5 (requires adding disks first)
sudo mdadm --add /dev/md0 /dev/sdd
sudo mdadm --grow /dev/md0 --level=5 --raid-devices=3

# Convert RAID 5 to RAID 6
sudo mdadm --grow /dev/md0 --level=6 --raid-devices=4 --add /dev/sde

# Note: Always backup data before conversion!
```

Q: How do I recover from a RAID failure?

A: Recovery steps depend on the failure type:

```
# For single disk failure in RAID 1/5/6:
# 1. Replace failed disk
sudo mdadm --remove /dev/md0 /dev/sdb  # Remove failed disk
# 2. Add new disk
sudo mdadm --add /dev/md0 /dev/sdb

# For array corruption:
# 1. Stop array
sudo mdadm --stop /dev/md0
# 2. Assemble with force
sudo mdadm --assemble --force /dev/md0 /dev/sdb /dev/sdc

# For complete array loss:
# 1. Try to recreate array with same parameters
# 2. Use data recovery tools
# 3. Restore from backup
```

Q: How do I monitor RAID health automatically?

A: Set up automated monitoring:

```
# Configure mdadm monitoring
echo "MAILADDR admin@example.com" | sudo tee -a /etc/mdadm/mdadm.conf

# Test email notification
sudo mdadm --monitor --test /dev/md0

# Set up continuous monitoring service
sudo systemctl enable mdmonitor
sudo systemctl start mdmonitor
```

Coding Examples

Python RAID Monitoring Script

```
#!/usr/bin/env python3
"""
RAID monitoring and management script for Ubuntu 22.04
"""
import subprocess
import re
import json
import time
import smtplib
from email.mime.text import MIMEText
from datetime import datetime

class RAIDMonitor:
    def __init__(self):
```

```
        self.raid_devices = self.discover_raid_devices()

    def discover_raid_devices(self):
        """Discover all RAID devices on the system"""
        try:
            result = subprocess.run(['cat', '/proc/mdstat'],
                                    capture_output=True, text=True)

            devices = []
            for line in result.stdout.split('\\n'):
                if line.startswith('md'):
                    device_name = line.split()[0]
                    devices.append(f'/dev/{device_name}')

            return devices
        except Exception as e:
            print(f"Error discovering RAID devices: {e}")
            return []

    def get_raid_status(self, device):
        """Get detailed status of a RAID device"""
        try:
            result = subprocess.run(['mdadm', '--detail', device],
                                    capture_output=True, text=True)

            if result.returncode != 0:
                return {'error': f'Failed to get status for {device}'}

            status = {'device': device, 'healthy': True, 'details': {}}

            for line in result.stdout.split('\\n'):
                line = line.strip()

                if 'State :' in line:
                    state = line.split(':', 1)[1].strip()
                    status['details']['state'] = state
                    if 'clean' not in state.lower():
                        status['healthy'] = False

                elif 'RAID Level :' in line:
                    status['details']['level'] = line.split(':', 1)[1].strip()

                elif 'Array Size :' in line:
                    status['details']['size'] = line.split(':', 1)[1].strip()

                elif 'Used Dev Size :' in line:
                    status['details']['used_size'] = line.split(':', 1)[1].strip()

                elif 'Active Devices :' in line:
                    status['details']['active_devices'] = int(line.split(':', 1)
[1].strip())

                elif 'Working Devices :' in line:
                    status['details']['working_devices'] = int(line.split(':', 1)
[1].strip())

                elif 'Failed Devices :' in line:
                    failed = int(line.split(':', 1)[1].strip())
                    status['details']['failed_devices'] = failed
                    if failed > 0:
                        status['healthy'] = False

            return status

        except Exception as e:
```

```
            return {'device': device, 'error': str(e)}

    def get_disk_health(self, device):
        """Check individual disk health using SMART"""
        try:
            result = subprocess.run(['smartctl', '-H', device],
                                    capture_output=True, text=True)

            if 'PASSED' in result.stdout:
                return {'device': device, 'smart_status': 'PASSED', 'healthy': True}
            else:
                return {'device': device, 'smart_status': 'FAILED', 'healthy': False}

        except Exception as e:
            return {'device': device, 'error': str(e)}

    def check_array_sync(self, device):
        """Check if array is currently syncing/rebuilding"""
        try:
            with open('/proc/mdstat', 'r') as f:
                content = f.read()

            device_name = device.split('/')[-1]

            for line in content.split('\\n'):
                if device_name in line:
                    if 'resync' in line or 'recovery' in line or 'rebuild' in line:
                        # Extract progress if available
                        progress_match = re.search(r'\\[([^\\]]+)\\]', line)
                        if progress_match:
                            return {
                                'syncing': True,
                                'progress': progress_match.group(1),
                                'status': line.strip()
                            }
                        return {'syncing': True, 'status': line.strip()}

            return {'syncing': False}

        except Exception as e:
            return {'error': str(e)}

    def generate_report(self):
        """Generate comprehensive RAID status report"""
        report = {
            'timestamp': datetime.now().isoformat(),
            'hostname': subprocess.run(['hostname'], capture_output=True,
text=True).stdout.strip(),
            'raid_devices': [],
            'overall_health': True
        }

        for device in self.raid_devices:
            device_status = self.get_raid_status(device)
            sync_status = self.check_array_sync(device)

            device_report = {
                'device': device,
                'status': device_status,
                'sync': sync_status
            }

            if not device_status.get('healthy', False):
                report['overall_health'] = False
```

```
            report['raid_devices'].append(device_report)

        return report

    def send_alert(self, message, subject="RAID Alert"):
        """Send email alert (requires SMTP configuration)"""
        try:
            # Configure SMTP settings
            smtp_server = "localhost"
            smtp_port = 587
            from_email = "raid-monitor@localhost"
            to_email = "admin@localhost"

            msg = MIMEText(message)
            msg['Subject'] = subject
            msg['From'] = from_email
            msg['To'] = to_email

            server = smtplib.SMTP(smtp_server, smtp_port)
            server.send_message(msg)
            server.quit()

            return True
        except Exception as e:
            print(f"Failed to send alert: {e}")
            return False

    def monitor_continuous(self, interval=300):
        """Continuous monitoring with specified interval (seconds)"""
        print(f"Starting RAID monitoring (checking every {interval} seconds)")

        while True:
            try:
                report = self.generate_report()

                if not report['overall_health']:
                    alert_message = f"RAID Health Alert - {report['hostname']}\\n\\n"
                    alert_message += json.dumps(report, indent=2)

                    print(f"ALERT: RAID issues detected at {report['timestamp']}")
                    self.send_alert(alert_message, "RAID Health Alert")
                else:
                    print(f"All RAID devices healthy at {report['timestamp']}")

                time.sleep(interval)

            except KeyboardInterrupt:
                print("\\nMonitoring stopped by user")
                break
            except Exception as e:
                print(f"Error during monitoring: {e}")
                time.sleep(interval)

# Example usage and CLI interface
if __name__ == "__main__":
    import argparse

    parser = argparse.ArgumentParser(description='RAID Monitoring Tool')
    parser.add_argument('--monitor', action='store_true', help='Start continuous
monitoring')
    parser.add_argument('--report', action='store_true', help='Generate one-time
report')
    parser.add_argument('--device', help='Check specific device')
    parser.add_argument('--interval', type=int, default=300, help='Monitoring interval
in seconds')
```

```
    args = parser.parse_args()

    monitor = RAIDMonitor()

    if args.monitor:
        monitor.monitor_continuous(args.interval)
    elif args.report:
        report = monitor.generate_report()
        print(json.dumps(report, indent=2))
    elif args.device:
        status = monitor.get_raid_status(args.device)
        print(json.dumps(status, indent=2))
    else:
        # Default: show brief status
        for device in monitor.raid_devices:
            status = monitor.get_raid_status(device)
            health = "HEALTHY" if status.get('healthy', False) else "UNHEALTHY"
            print(f"{device}: {health}")
```

Bash RAID Management Script

```
#!/bin/bash
# raid_manager.sh - Comprehensive RAID management for Ubuntu 22.04

# Configuration
CONFIG_FILE="/etc/raid_manager.conf"
LOG_FILE="/var/log/raid_manager.log"
ALERT_EMAIL="admin@localhost"

# Colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
BLUE='\033[0;34m'
NC='\033[0m'

# Logging function
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

# Check if running as root
check_root() {
    if [[ $EUID -ne 0 ]]; then
        echo -e "${RED}This script must be run as root${NC}"
        exit 1
    fi
}

# Install required packages
install_dependencies() {
    echo -e "${BLUE}Installing RAID dependencies...${NC}"

    apt update
    apt install -y mdadm smartmontools parted gdisk mail-utils

    # Enable mdadm monitoring
    systemctl enable mdmonitor
    systemctl start mdmonitor

    log_message "RAID dependencies installed"
}
```

```
# Discover RAID devices
discover_raids() {
    echo -e "${BLUE}=== Discovered RAID Devices ===${NC}"

    if [[ -f /proc/mdstat ]]; then
        cat /proc/mdstat
        echo ""

        # List detailed information for each device
        for device in /dev/md*; do
            if [[ -b "$device" ]]; then
                echo -e "${GREEN}Details for $device:${NC}"
                mdadm --detail "$device" 2>/dev/null | head -20
                echo ""
            fi
        done
    else
        echo "No RAID devices found"
    fi
}

# Create RAID array
create_raid() {
    local level="$1"
    local devices="$2"
    local array_name="$3"

    if [[ -z "$level" ]] || [[ -z "$devices" ]]; then
        echo "Usage: create_raid <level> <device1,device2,...> [array_name]"
        echo "Example: create_raid 1 /dev/sdb,/dev/sdc myraid"
        return 1
    fi

    local device_array=(${devices//,/ })
    local device_count=${#device_array[@]}
    local raid_device="/dev/md${array_name:-$(date +%s)}"

    echo -e "${YELLOW}Creating RAID $level with $device_count devices${NC}"

    # Validate RAID level and device count
    case "$level" in
        "0")
            if [[ $device_count -lt 2 ]]; then
                echo -e "${RED}RAID 0 requires at least 2 devices${NC}"
                return 1
            fi
            ;;
        "1")
            if [[ $device_count -lt 2 ]]; then
                echo -e "${RED}RAID 1 requires at least 2 devices${NC}"
                return 1
            fi
            ;;
        "5")
            if [[ $device_count -lt 3 ]]; then
                echo -e "${RED}RAID 5 requires at least 3 devices${NC}"
                return 1
            fi
            ;;
        "6")
            if [[ $device_count -lt 4 ]]; then
                echo -e "${RED}RAID 6 requires at least 4 devices${NC}"
                return 1
            fi
            ;;
```

```
        "10")
            if [[ $device_count -lt 4 ]] || [[ $((device_count % 2)) -ne 0 ]]; then
                echo -e "${RED}RAID 10 requires at least 4 devices (even number)${NC}"
                return 1
            fi
            ;;
        *)
            echo -e "${RED}Unsupported RAID level: $level${NC}"
            return 1
            ;;
    esac

    # Prepare devices
    echo "Preparing devices..."
    for device in "${device_array[@]}"; do
        echo "Wiping $device..."
        wipefs -a "$device"

        # Zero superblock if exists
        mdadm --zero-superblock "$device" 2>/dev/null
    done

    # Create RAID array
    echo "Creating RAID array..."
    if mdadm --create --verbose "$raid_device" \\
        --level="$level" \\
        --raid-devices="$device_count" \\
        "${device_array[@]}"; then

        echo -e "${GREEN}RAID array created successfully: $raid_device${NC}"

        # Save configuration
        mdadm --detail --scan >> /etc/mdadm/mdadm.conf
        update-initramfs -u

        log_message "Created RAID $level array $raid_device with devices: $devices"

        # Wait for array to synchronize
        echo "Waiting for initial synchronization..."
        while grep -q "resync" /proc/mdstat; do
            echo -n "."
            sleep 5
        done
        echo ""
        echo -e "${GREEN}Initial synchronization complete${NC}"

    else
        echo -e "${RED}Failed to create RAID array${NC}"
        return 1
    fi
}

# Monitor RAID health
monitor_raid_health() {
    echo -e "${BLUE}=== RAID Health Monitor ===${NC}"

    local unhealthy_found=false

    # Check each RAID device
    for device in /dev/md*; do
        if [[ -b "$device" ]]; then
            echo "Checking $device..."

            # Get RAID status
            local status=$(mdadm --detail "$device" 2>/dev/null | grep "State :" | awk
```

```
'{print $3}')

            if [[ "$status" == "clean" ]]; then
                echo -e "${GREEN}$device: Healthy ($status)${NC}"
            else
                echo -e "${RED}$device: Unhealthy ($status)${NC}"
                unhealthy_found=true

                # Get detailed information
                mdadm --detail "$device"
            fi

            # Check for rebuild/resync
            if grep -q "$(basename $device)" /proc/mdstat; then
                local sync_info=$(grep "$(basename $device)" /proc/mdstat | grep -E "
(resync|rebuild|recovery)")
                if [[ -n "$sync_info" ]]; then
                    echo -e "${YELLOW}$device: $sync_info${NC}"
                fi
            fi

            echo ""
        fi
    done

    # Check individual disk health
    echo -e "${BLUE}=== Individual Disk Health ===${NC}"
    for device in /dev/sd[a-z]; do
        if [[ -b "$device" ]]; then
            local smart_status=$(smartctl -H "$device" 2>/dev/null | grep "SMART
overall-health" | awk '{print $6}')
            if [[ "$smart_status" == "PASSED" ]]; then
                echo -e "${GREEN}$device: SMART PASSED${NC}"
            elif [[ "$smart_status" == "FAILED" ]]; then
                echo -e "${RED}$device: SMART FAILED${NC}"
                unhealthy_found=true
            fi
        fi
    done

    # Send alert if issues found
    if [[ "$unhealthy_found" == "true" ]]; then
        log_message "RAID health issues detected"
        send_alert "RAID health issues detected on $(hostname)"
    fi
}

# Manage RAID device
manage_raid() {
    local action="$1"
    local device="$2"
    local target="$3"

    case "$action" in
        "add")
            if [[ -z "$device" ]] || [[ -z "$target" ]]; then
                echo "Usage: manage_raid add <raid_device> <new_disk>"
                return 1
            fi

            echo -e "${YELLOW}Adding $target to $device${NC}"
            if mdadm --add "$device" "$target"; then
                echo -e "${GREEN}Disk added successfully${NC}"
                log_message "Added disk $target to $device"
            else
```

```
            echo -e "${RED}Failed to add disk${NC}"
        fi
        ;;

    "remove")
        if [[ -z "$device" ]] || [[ -z "$target" ]]; then
            echo "Usage: manage_raid remove <raid_device> <disk>"
            return 1
        fi

        echo -e "${YELLOW}Removing $target from $device${NC}"
        if mdadm --remove "$device" "$target"; then
            echo -e "${GREEN}Disk removed successfully${NC}"
            log_message "Removed disk $target from $device"
        else
            echo -e "${RED}Failed to remove disk${NC}"
        fi
        ;;

    "fail")
        if [[ -z "$device" ]] || [[ -z "$target" ]]; then
            echo "Usage: manage_raid fail <raid_device> <disk>"
            return 1
        fi

        echo -e "${YELLOW}Marking $target as failed in $device${NC}"
        if mdadm --fail "$device" "$target"; then
            echo -e "${GREEN}Disk marked as failed${NC}"
            log_message "Marked disk $target as failed in $device"
        else
            echo -e "${RED}Failed to mark disk as failed${NC}"
        fi
        ;;

    "replace")
        if [[ -z "$device" ]] || [[ -z "$target" ]]; then
            echo "Usage: manage_raid replace <raid_device> <old_disk> <new_disk>"
            echo "Note: old_disk should be failed first"
            return 1
        fi

        local new_disk="$4"
        if [[ -z "$new_disk" ]]; then
            echo "New disk not specified"
            return 1
        fi

        echo -e "${YELLOW}Replacing $target with $new_disk in $device${NC}"

        # Remove failed disk
        mdadm --remove "$device" "$target"

        # Add new disk
        if mdadm --add "$device" "$new_disk"; then
            echo -e "${GREEN}Disk replacement initiated${NC}"
            log_message "Replaced disk $target with $new_disk in $device"
        else
            echo -e "${RED}Failed to add replacement disk${NC}"
        fi
        ;;

    *)
        echo "Available actions: add, remove, fail, replace"
        ;;
esac
```

```
}

# Send email alert
send_alert() {
    local message="$1"

    if command -v mail &> /dev/null; then
        echo "$message" | mail -s "RAID Alert - $(hostname)" "$ALERT_EMAIL"
        log_message "Alert sent: $message"
    else
        log_message "Alert (mail not configured): $message"
    fi
}

# Performance test
test_raid_performance() {
    local device="$1"
    local test_file="$2"

    if [[ -z "$device" ]]; then
        echo "Usage: test_raid_performance <device> [test_file]"
        return 1
    fi

    local mount_point="/mnt/raid_test"
    local test_path="${test_file:-$mount_point/test_file}"

    echo -e "${BLUE}=== RAID Performance Test ===${NC}"

    # Mount if not already mounted
    if ! mountpoint -q "$mount_point"; then
        mkdir -p "$mount_point"
        mount "$device" "$mount_point"
    fi

    echo "Testing write performance..."
    local write_speed=$(dd if=/dev/zero of="$test_path" bs=1M count=1024 2>&1 | grep
"MB/s" | awk '{print $(NF-1) " " $NF}')
    echo "Write speed: $write_speed"

    echo "Testing read performance..."
    local read_speed=$(dd if="$test_path" of=/dev/null bs=1M 2>&1 | grep "MB/s" | awk
'{print $(NF-1) " " $NF}')
    echo "Read speed: $read_speed"

    # Cleanup
    rm -f "$test_path"

    log_message "Performance test for $device - Write: $write_speed, Read:
$read_speed"
}

# Main menu
case "${1:-help}" in
    "install")
        check_root
        install_dependencies
        ;;
    "discover")
        discover_raids
        ;;
    "create")
        check_root
        create_raid "$2" "$3" "$4"
        ;;
```

```
    "monitor")
        monitor_raid_health
        ;;
    "manage")
        check_root
        manage_raid "$2" "$3" "$4" "$5"
        ;;
    "performance")
        test_raid_performance "$2" "$3"
        ;;
    "help"|*)
        echo "RAID Manager for Ubuntu 22.04"
        echo "Usage: $0 [command] [options]"
        echo ""
        echo "Commands:"
        echo "  install                          - Install RAID dependencies"
        echo "  discover                         - Discover existing RAID devices"
        echo "  create <level> <devices> [name]  - Create new RAID array"
        echo "  monitor                          - Check RAID health"
        echo "  manage <action> <device> <disk>  - Manage RAID devices"
        echo "  performance <device> [file]      - Test RAID performance"
        echo ""
        echo "RAID Levels: 0, 1, 5, 6, 10"
        echo "Manage Actions: add, remove, fail, replace"
        echo ""
        echo "Examples:"
        echo "  $0 create 1 /dev/sdb,/dev/sdc mirror1"
        echo "  $0 create 5 /dev/sdb,/dev/sdc,/dev/sdd stripe1"
        echo "  $0 manage add /dev/md0 /dev/sde"
        echo "  $0 manage replace /dev/md0 /dev/sdb /dev/sdf"
        echo "  $0 performance /dev/md0"
        ;;
esac
```

Best Practices for RAID Implementation

Planning and Design

1. **Capacity Planning**:
    - Calculate usable capacity for each RAID level
    - Plan for future expansion
    - Consider spare drives for automatic rebuilds
    - Account for rebuild times
2. **Performance Considerations**:
    - Match disk speeds and sizes
    - Use appropriate chunk sizes for workload
    - Consider I/O patterns
    - Plan for concurrent operations

Hardware Considerations

1. **Disk Selection**:

```
# Check disk specifications
sudo hdparm -I /dev/sdb | grep -E "(Model|Serial|Capacity)"

# Verify disks are identical
for disk in /dev/sd[b-d]; do
    echo "=== $disk ==="
    sudo smartctl -i $disk | grep -E "(Device Model|Serial Number|User Capacity)"
done
```

2. **Controller Considerations**:
    - Hardware RAID vs Software RAID
    - Battery-backed write cache
    - Multiple controller support
    - Hot-swap capability

Monitoring and Maintenance

1. **Automated Monitoring**:

```
# Set up email notifications
echo "MAILADDR admin@example.com" >> /etc/mdadm/mdadm.conf

# Configure monitoring daemon
systemctl enable mdmonitor

# Test notifications
mdadm --monitor --test /dev/md0
```

2. **Regular Maintenance**:

```
# Schedule regular RAID checks
echo "0 1 1 * * root echo check > /sys/block/md0/md/sync_action" >> /etc/crontab

# Monitor rebuild progress
watch cat /proc/mdstat

# Check SMART status regularly
for disk in /dev/sd[a-z]; do
    smartctl -a $disk | grep -E "(Health|Temperature|Error)"
done
```

Recovery and Disaster Planning

1. **Backup Strategy**:
    - RAID is not a backup solution
    - Implement 3-2-1 backup rule
    - Test restoration procedures
    - Document recovery processes

2. **Disaster Recovery**:

```
# Save RAID configuration
mdadm --detail --scan > /etc/mdadm/mdadm.conf.backup

# Create recovery documentation
```

```
cat > /root/raid_recovery.txt << 'EOF'
RAID Recovery Procedures:

1. Boot from rescue media
2. Install mdadm: apt install mdadm
3. Scan for arrays: mdadm --assemble --scan
4. Force assembly if needed: mdadm --assemble --force /dev/md0 /dev/sdb /dev/sdc
5. Mount and check data
EOF
```

Network Storage

Understanding Network Storage

Network storage allows multiple systems to access shared storage resources over a network. This technology is essential for modern IT environments, enabling centralized data management, improved collaboration, and efficient resource utilization.

Types of Network Storage

Network Attached Storage (NAS)

Characteristics: * File-level storage access * Uses standard network protocols (NFS, SMB/CIFS, FTP) * Easy to deploy and manage * Suitable for file sharing and collaboration

Common Protocols: * **NFS (Network File System)**: Unix/Linux native protocol * **SMB/CIFS**: Windows native, also supported by Linux * **FTP/SFTP**: File transfer protocols * **HTTP/WebDAV**: Web-based file access

Storage Area Network (SAN)

Characteristics: * Block-level storage access * High-performance dedicated network * Uses specialized protocols (iSCSI, Fibre Channel) * Enterprise-level performance and reliability

Common Protocols: * **iSCSI**: SCSI over IP networks * **Fibre Channel**: High-speed dedicated network * **FCoE**: Fibre Channel over Ethernet * **InfiniBand**: High-performance computing networks

Cloud Storage

Characteristics: * Storage as a Service (STaaS) * Scalable and elastic * Geographic distribution * Pay-per-use model

Common Services: * **Object Storage**: Amazon S3, Google Cloud Storage, Azure Blob * **File Storage**: Amazon EFS, Google Filestore, Azure Files * **Block Storage**: Amazon EBS, Google Persistent Disk, Azure Disk

Setting Up NFS on Ubuntu 22.04

NFS Server Configuration

```
# Install NFS server
sudo apt update
sudo apt install nfs-kernel-server
```

```
# Create shared directories
sudo mkdir -p /srv/nfs/shared
sudo mkdir -p /srv/nfs/public
sudo mkdir -p /srv/nfs/private

# Set permissions
sudo chown nobody:nogroup /srv/nfs/shared
sudo chown nobody:nogroup /srv/nfs/public
sudo chmod 755 /srv/nfs/shared
sudo chmod 755 /srv/nfs/public

# Configure exports
sudo nano /etc/exports

# Add export configurations
cat >> /etc/exports << 'EOF'
# NFS exports configuration
/srv/nfs/shared    192.168.1.0/24(rw,sync,no_subtree_check)
/srv/nfs/public    *(ro,sync,no_subtree_check)
/srv/nfs/private   192.168.1.100(rw,sync,no_subtree_check,no_root_squash)
EOF

# Export the shares
sudo exportfs -a

# Restart NFS service
sudo systemctl restart nfs-kernel-server
sudo systemctl enable nfs-kernel-server

# Check NFS status
sudo systemctl status nfs-kernel-server
sudo exportfs -v
```

NFS Client Configuration

```
# Install NFS client
sudo apt update
sudo apt install nfs-common

# Create mount points
sudo mkdir -p /mnt/nfs/shared
sudo mkdir -p /mnt/nfs/public

# Test NFS connectivity
showmount -e 192.168.1.10

# Mount NFS shares
sudo mount -t nfs 192.168.1.10:/srv/nfs/shared /mnt/nfs/shared
sudo mount -t nfs 192.168.1.10:/srv/nfs/public /mnt/nfs/public

# Verify mounts
df -h | grep nfs

# Add to fstab for persistent mounting
cat >> /etc/fstab << 'EOF'
# NFS mounts
192.168.1.10:/srv/nfs/shared  /mnt/nfs/shared  nfs  defaults,_netdev  0  0
192.168.1.10:/srv/nfs/public  /mnt/nfs/public  nfs  ro,defaults,_netdev  0  0
EOF

# Test fstab entries
sudo umount /mnt/nfs/shared
sudo mount -a
```

Setting Up Samba (SMB/CIFS) on Ubuntu 22.04

Samba Server Configuration

```
# Install Samba
sudo apt update
sudo apt install samba samba-common-bin

# Create shared directories
sudo mkdir -p /srv/samba/shared
sudo mkdir -p /srv/samba/public
sudo mkdir -p /srv/samba/users

# Set permissions
sudo chown root:sambashare /srv/samba/shared
sudo chown root:sambashare /srv/samba/public
sudo chown root:sambashare /srv/samba/users
sudo chmod 2775 /srv/samba/shared
sudo chmod 2775 /srv/samba/public
sudo chmod 2775 /srv/samba/users

# Backup original configuration
sudo cp /etc/samba/smb.conf /etc/samba/smb.conf.backup

# Configure Samba
cat >> /etc/samba/smb.conf << 'EOF'

[shared]
    comment = Shared Files
    path = /srv/samba/shared
    browseable = yes
    writable = yes
    guest ok = no
    valid users = @sambashare
    create mask = 0664
    directory mask = 2775

[public]
    comment = Public Files
    path = /srv/samba/public
    browseable = yes
    writable = yes
    guest ok = yes
    read only = no
    create mask = 0755

[users]
    comment = User Directories
    path = /srv/samba/users/%S
    browseable = no
    writable = yes
    valid users = %S
    create mask = 0644
    directory mask = 0755
EOF

# Create Samba user
sudo useradd -M -s /usr/sbin/nologin sambauser
sudo usermod -aG sambashare sambauser
sudo smbpasswd -a sambauser

# Restart Samba services
sudo systemctl restart smbd nmbd
sudo systemctl enable smbd nmbd
```

```
# Check Samba status
sudo systemctl status smbd
testparm
```

Samba Client Configuration

```
# Install CIFS utilities
sudo apt install cifs-utils

# Create mount points
sudo mkdir -p /mnt/samba/shared
sudo mkdir -p /mnt/samba/public

# Mount Samba shares
sudo mount -t cifs //192.168.1.10/shared /mnt/samba/shared -o username=sambauser

# Create credentials file for automatic mounting
cat > ~/.smbcredentials << 'EOF'
username=sambauser
password=your_password
domain=workgroup
EOF

chmod 600 ~/.smbcredentials

# Add to fstab
cat >> /etc/fstab << 'EOF'
# Samba mounts
//192.168.1.10/shared  /mnt/samba/shared  cifs
credentials=/home/username/.smbcredentials,uid=1000,gid=1000,_netdev  0  0
//192.168.1.10/public  /mnt/samba/public  cifs  guest,uid=1000,gid=1000,_netdev  0  0
EOF
```

Setting Up iSCSI on Ubuntu 22.04

iSCSI Target Server Configuration

```
# Install iSCSI target
sudo apt update
sudo apt install tgt

# Create storage backing file
sudo mkdir -p /srv/iscsi
sudo dd if=/dev/zero of=/srv/iscsi/disk1.img bs=1M count=10240

# Configure iSCSI target
cat > /etc/tgt/conf.d/iscsi-target.conf << 'EOF'
# iSCSI Target Configuration
<target iqn.2024-01.com.example:storage.disk1>
    backing-store /srv/iscsi/disk1.img
    # Allow access from specific initiators
    initiator-address 192.168.1.0/24
    # Authentication (optional)
    incominguser username password
</target>
EOF

# Restart target service
sudo systemctl restart tgt
sudo systemctl enable tgt

# Check target status
```

```
sudo tgtadm --mode target --op show
```

iSCSI Initiator Configuration

```
# Install iSCSI initiator
sudo apt update
sudo apt install open-iscsi

# Configure initiator name
sudo nano /etc/iscsi/initiatorname.iscsi
# Set: InitiatorName=iqn.2024-01.com.example:client1

# Configure CHAP authentication (if required)
sudo nano /etc/iscsi/iscsid.conf
# Uncomment and set:
# node.session.auth.authmethod = CHAP
# node.session.auth.username = username
# node.session.auth.password = password

# Discover targets
sudo iscsiadm -m discovery -t sendtargets -p 192.168.1.10

# Login to target
sudo iscsiadm -m node --login

# Check connected sessions
sudo iscsiadm -m session

# Format and mount iSCSI disk
sudo fdisk /dev/sdb  # Create partition
sudo mkfs.ext4 /dev/sdb1
sudo mkdir /mnt/iscsi
sudo mount /dev/sdb1 /mnt/iscsi
```

Cloud Storage Integration

Rclone Configuration

```
# Install rclone
sudo apt update
sudo apt install rclone

# Configure cloud storage
rclone config

# Example: Mount Google Drive
mkdir ~/GoogleDrive
rclone mount gdrive: ~/GoogleDrive --daemon

# Example: Sync to cloud storage
rclone sync /home/user/Documents gdrive:Documents

# Create systemd service for persistent mounting
cat > ~/.config/systemd/user/rclone-gdrive.service << 'EOF'
[Unit]
Description=Google Drive mount
After=network.target

[Service]
Type=notify
ExecStart=/usr/bin/rclone mount gdrive: %h/GoogleDrive --vfs-cache-mode writes
ExecStop=/bin/fusermount -u %h/GoogleDrive
Restart=always
```

```
RestartSec=10

[Install]
WantedBy=default.target
EOF

systemctl --user enable rclone-gdrive.service
systemctl --user start rclone-gdrive.service
```

S3-Compatible Storage

```
# Install s3fs
sudo apt install s3fs

# Create credentials file
echo "access_key:secret_key" > ~/.passwd-s3fs
chmod 600 ~/.passwd-s3fs

# Mount S3 bucket
mkdir ~/s3bucket
s3fs mybucket ~/s3bucket -o passwd_file=~/.passwd-s3fs -o url=https://s3.amazonaws.com

# Add to fstab for persistent mounting
echo "s3fs#mybucket /home/user/s3bucket fuse _netdev,passwd_file=/home/user/.passwd-
s3fs,url=https://s3.amazonaws.com 0 0" >> /etc/fstab
```

Network Storage Performance Optimization

NFS Performance Tuning

```
# Optimize NFS mount options
sudo mount -t nfs -o rsize=32768,wsize=32768,hard,intr,tcp
192.168.1.10:/srv/nfs/shared /mnt/nfs/shared

# Configure NFS daemon threads
sudo nano /etc/default/nfs-kernel-server
# Set: RPCNFSDCOUNT=16

# Optimize network buffer sizes
echo "net.core.rmem_max = 16777216" >> /etc/sysctl.conf
echo "net.core.wmem_max = 16777216" >> /etc/sysctl.conf
sudo sysctl -p
```

Samba Performance Tuning

```
# Add performance options to smb.conf
cat >> /etc/samba/smb.conf << 'EOF'
[global]
    # Performance tuning
    socket options = TCP_NODELAY IPTOS_LOWDELAY SO_RCVBUF=65536 SO_SNDBUF=65536
    read raw = yes
    write raw = yes
    max xmit = 65535
    dead time = 15
    getwd cache = yes
EOF

sudo systemctl restart smbd
```

iSCSI Performance Tuning

```
# Optimize iSCSI parameters
```

```
echo "node.session.iscsi.InitialR2T = No" >> /etc/iscsi/iscsid.conf
echo "node.session.iscsi.ImmediateData = Yes" >> /etc/iscsi/iscsid.conf
echo "node.session.iscsi.FirstBurstLength = 262144" >> /etc/iscsi/iscsid.conf
echo "node.session.iscsi.MaxBurstLength = 16776192" >> /etc/iscsi/iscsid.conf
echo "node.conn[0].iscsi.MaxRecvDataSegmentLength = 262144" >> /etc/iscsi/iscsid.conf

sudo systemctl restart iscsid
```

Frequently Asked Questions

Q: Which network storage protocol should I choose?

A: Choose based on your requirements:

```
Use Case                    | Recommended Protocol | Reason
----------------------------|----------------------|-----------------
Linux-only environment     | NFS                 | Native performance
Mixed OS environment       | SMB/CIFS            | Universal compatibility
High-performance storage    | iSCSI               | Block-level access
Simple file sharing         | FTP/SFTP            | Easy setup
Cloud storage              | Object storage APIs  | Scalability
Database storage           | iSCSI or FC         | Low latency
```

Q: How do I troubleshoot NFS connection issues?

A: Follow these troubleshooting steps:

```
# Check NFS service status
sudo systemctl status nfs-kernel-server

# Check exports
sudo exportfs -v

# Test connectivity
telnet nfs-server-ip 2049

# Check firewall
sudo ufw status
# Open NFS ports if needed
sudo ufw allow from 192.168.1.0/24 to any port 2049

# Check mount on client
showmount -e nfs-server-ip

# Debug mount issues
sudo mount -t nfs -v nfs-server-ip:/path /mnt/point
```

Q: How can I secure network storage?

A: Implement these security measures:

```
# 1. Use VPN for remote access
sudo apt install openvpn

# 2. Enable firewall
sudo ufw enable
sudo ufw allow from 192.168.1.0/24 to any port 2049  # NFS
sudo ufw allow from 192.168.1.0/24 to any port 445   # SMB

# 3. Use authentication
# For NFS: Configure Kerberos
# For SMB: Use strong passwords and encryption
```

```
# 4. Enable encryption
# For SMB, add to smb.conf:
echo "smb encrypt = required" >> /etc/samba/smb.conf

# 5. Regular security updates
sudo apt update && sudo apt upgrade
```

Q: How do I monitor network storage performance?

A: Use these monitoring tools:

```
# Monitor network I/O
iftop -i eth0

# Monitor NFS statistics
nfsstat -c  # Client stats
nfsstat -s  # Server stats

# Monitor iSCSI performance
iostat -x 1

# Monitor Samba connections
sudo smbstatus

# Create monitoring script
cat > monitor_network_storage.sh << 'EOF'
#!/bin/bash

echo "=== Network Storage Performance Monitor ==="
echo "Date: $(date)"
echo ""

echo "NFS Statistics:"
nfsstat -c 2>/dev/null || echo "NFS client not active"
echo ""

echo "Network Interface Statistics:"
cat /proc/net/dev | grep -E "(eth0|ens|enp)"
echo ""

echo "Active Network Connections:"
ss -tuln | grep -E "(2049|445|3260)"  # NFS, SMB, iSCSI
EOF

chmod +x monitor_network_storage.sh
```

Coding Examples

Python Network Storage Monitor

```
#!/usr/bin/env python3
"""
Network storage monitoring script for Ubuntu 22.04
"""
import subprocess
import psutil
import time
import json
from datetime import datetime

class NetworkStorageMonitor:
    def __init__(self):
```

```
    self.start_time = time.time()
    self.interfaces = self.get_network_interfaces()

def get_network_interfaces(self):
    """Get list of network interfaces"""
    return list(psutil.net_if_addrs().keys())

def check_nfs_mounts(self):
    """Check NFS mount status"""
    try:
        result = subprocess.run(['mount', '-t', 'nfs'],
                                capture_output=True, text=True)

        mounts = []
        for line in result.stdout.strip().split('\\n'):
            if line:
                parts = line.split()
                if len(parts) >= 6:
                    mounts.append({
                        'server': parts[0],
                        'mountpoint': parts[2],
                        'options': parts[5]
                    })

        return {'status': 'success', 'mounts': mounts}
    except Exception as e:
        return {'status': 'error', 'message': str(e)}

def check_smb_mounts(self):
    """Check SMB/CIFS mount status"""
    try:
        result = subprocess.run(['mount', '-t', 'cifs'],
                                capture_output=True, text=True)

        mounts = []
        for line in result.stdout.strip().split('\\n'):
            if line:
                parts = line.split()
                if len(parts) >= 6:
                    mounts.append({
                        'server': parts[0],
                        'mountpoint': parts[2],
                        'options': parts[5]
                    })

        return {'status': 'success', 'mounts': mounts}
    except Exception as e:
        return {'status': 'error', 'message': str(e)}

def check_iscsi_sessions(self):
    """Check iSCSI session status"""
    try:
        result = subprocess.run(['iscsiadm', '-m', 'session'],
                                capture_output=True, text=True)

        sessions = []
        for line in result.stdout.strip().split('\\n'):
            if line and 'tcp:' in line:
                sessions.append(line.strip())

        return {'status': 'success', 'sessions': sessions}
    except Exception as e:
        return {'status': 'error', 'message': str(e)}

def get_network_stats(self):
```

```
    """Get network interface statistics"""
    stats = {}
    net_io = psutil.net_io_counters(pernic=True)

    for interface, counters in net_io.items():
        stats[interface] = {
            'bytes_sent': counters.bytes_sent,
            'bytes_recv': counters.bytes_recv,
            'packets_sent': counters.packets_sent,
            'packets_recv': counters.packets_recv,
            'errors_in': counters.errin,
            'errors_out': counters.errout,
            'drops_in': counters.dropin,
            'drops_out': counters.dropout
        }

    return stats

def check_service_status(self, service_name):
    """Check systemd service status"""
    try:
        result = subprocess.run(['systemctl', 'is-active', service_name],
                                capture_output=True, text=True)
        return result.stdout.strip()
    except Exception:
        return 'unknown'

def test_connectivity(self, host, port):
    """Test network connectivity to host:port"""
    try:
        import socket
        sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        sock.settimeout(5)
        result = sock.connect_ex((host, port))
        sock.close()
        return result == 0
    except Exception:
        return False

def generate_report(self):
    """Generate comprehensive network storage report"""
    report = {
        'timestamp': datetime.now().isoformat(),
        'uptime': time.time() - self.start_time,
        'services': {
            'nfs-server': self.check_service_status('nfs-kernel-server'),
            'smbd': self.check_service_status('smbd'),
            'nmbd': self.check_service_status('nmbd'),
            'iscsid': self.check_service_status('iscsid'),
            'tgt': self.check_service_status('tgt')
        },
        'mounts': {
            'nfs': self.check_nfs_mounts(),
            'smb': self.check_smb_mounts()
        },
        'iscsi': self.check_iscsi_sessions(),
        'network': self.get_network_stats()
    }

    return report

def monitor_performance(self, duration=60, interval=5):
    """Monitor network storage performance over time"""
    print(f"Monitoring network storage for {duration} seconds...")
```

```
        start_stats = self.get_network_stats()
        time.sleep(duration)
        end_stats = self.get_network_stats()

        performance = {}
        for interface in start_stats:
            if interface in end_stats:
                sent_diff = end_stats[interface]['bytes_sent'] -
start_stats[interface]['bytes_sent']
                recv_diff = end_stats[interface]['bytes_recv'] -
start_stats[interface]['bytes_recv']

                performance[interface] = {
                    'avg_send_rate_mbps': (sent_diff * 8) / (duration * 1024 * 1024),
                    'avg_recv_rate_mbps': (recv_diff * 8) / (duration * 1024 * 1024),
                    'total_sent_mb': sent_diff / (1024 * 1024),
                    'total_recv_mb': recv_diff / (1024 * 1024)
                }

        return performance

# Example usage
if __name__ == "__main__":
    import argparse

    parser = argparse.ArgumentParser(description='Network Storage Monitor')
    parser.add_argument('--report', action='store_true', help='Generate status
report')
    parser.add_argument('--monitor', type=int, help='Monitor performance for N
seconds')
    parser.add_argument('--json', action='store_true', help='Output in JSON format')

    args = parser.parse_args()

    monitor = NetworkStorageMonitor()

    if args.report:
        report = monitor.generate_report()
        if args.json:
            print(json.dumps(report, indent=2))
        else:
            print("Network Storage Status Report")
            print("=" * 40)
            print(f"Timestamp: {report['timestamp']}")
            print(f"\\nServices:")
            for service, status in report['services'].items():
                print(f"  {service}: {status}")

            print(f"\\nNFS Mounts: {len(report['mounts']['nfs']['mounts'])}")
            print(f"SMB Mounts: {len(report['mounts']['smb']['mounts'])}")
            print(f"iSCSI Sessions: {len(report['iscsi'].get('sessions', []))}")

    elif args.monitor:
        performance = monitor.monitor_performance(args.monitor)
        print(f"\\nNetwork Performance Summary ({args.monitor}s):")
        for interface, stats in performance.items():
            print(f"  {interface}:")
            print(f"    Average Send Rate: {stats['avg_send_rate_mbps']:.2f} Mbps")
            print(f"    Average Recv Rate: {stats['avg_recv_rate_mbps']:.2f} Mbps")

    else:
        parser.print_help()
```

Best Practices for Network Storage

Security Best Practices

1. **Network Segmentation**:
   - Use dedicated VLANs for storage traffic
   - Implement firewall rules
   - Regular security audits
2. **Authentication and Authorization**:
   - Strong passwords and multi-factor authentication
   - Regular credential rotation
   - Principle of least privilege
3. **Encryption**:
   - Encrypt data in transit
   - Encrypt data at rest
   - Use VPN for remote access

Performance Best Practices

1. **Network Optimization**:
   - Use dedicated high-speed networks
   - Optimize TCP window sizes
   - Enable jumbo frames where supported
2. **Protocol Selection**:
   - Choose appropriate protocols for workload
   - Optimize protocol-specific parameters
   - Consider multi-path configurations
3. **Monitoring and Maintenance**:
   - Regular performance monitoring
   - Proactive capacity planning
   - Scheduled maintenance windows

Volume Management

Volume management provides an abstraction layer between physical storage devices and file systems, enabling dynamic storage allocation, resizing, and advanced features like snapshots and encryption.

- Introduction to Volume Management

Introduction to Volume Management

Volume management systems allow you to:

- Combine multiple physical disks into logical volumes
- Resize volumes dynamically without downtime
- Create snapshots for backup and testing
- Implement storage encryption and compression

- Manage storage pools efficiently

Key Concepts

Physical Volumes (PV)

Physical storage devices or partitions

Volume Groups (VG)

Collections of physical volumes

Logical Volumes (LV)

Virtual partitions created from volume groups

Logical Volume Manager (LVM)

The primary volume management system in Linux

LVM (Logical Volume Manager)

Installation and Setup

```
# Install LVM tools on Ubuntu 22.04
sudo apt update
sudo apt install lvm2

# Check LVM installation
lvm version
```

Creating Physical Volumes

```
# Create physical volume from disk
sudo pvcreate /dev/sdb

# Create PV from partition
sudo pvcreate /dev/sdc1

# Display physical volumes
sudo pvdisplay
sudo pvs
```

Creating Volume Groups

```
# Create volume group from single PV
sudo vgcreate myvg /dev/sdb

# Create VG from multiple PVs
sudo vgcreate datavg /dev/sdb /dev/sdc

# Display volume groups
sudo vgdisplay
sudo vgs
```

Creating Logical Volumes

```
# Create LV with specific size
sudo lvcreate -L 10G -n mylv myvg
```

```
# Create LV using percentage of VG
sudo lvcreate -l 50%VG -n datalv datavg

# Create LV using all available space
sudo lvcreate -l 100%FREE -n homelv myvg

# Display logical volumes
sudo lvdisplay
sudo lvs
```

Formatting and Mounting LVM Volumes

```
# Format logical volume
sudo mkfs.ext4 /dev/myvg/mylv

# Create mount point
sudo mkdir /mnt/mylv

# Mount the volume
sudo mount /dev/myvg/mylv /mnt/mylv

# Add to /etc/fstab for persistent mounting
echo "/dev/myvg/mylv /mnt/mylv ext4 defaults 0 2" | sudo tee -a /etc/fstab
```

Resizing LVM Volumes

Extending Logical Volumes

```
# Extend LV size
sudo lvextend -L +5G /dev/myvg/mylv

# Extend LV to use all available space
sudo lvextend -l +100%FREE /dev/myvg/mylv

# Resize filesystem (ext4)
sudo resize2fs /dev/myvg/mylv

# For XFS filesystem
sudo xfs_growfs /mnt/mylv
```

Shrinking Logical Volumes

```
# Unmount the filesystem first
sudo umount /mnt/mylv

# Check filesystem
sudo e2fsck -f /dev/myvg/mylv

# Shrink filesystem first
sudo resize2fs /dev/myvg/mylv 8G

# Then shrink the logical volume
sudo lvreduce -L 8G /dev/myvg/mylv

# Remount
sudo mount /dev/myvg/mylv /mnt/mylv
```

LVM Snapshots

```
# Create snapshot
sudo lvcreate -L 2G -s -n mylv_snapshot /dev/myvg/mylv

# Mount snapshot for backup
```

```
sudo mkdir /mnt/snapshot
sudo mount /dev/myvg/mylv_snapshot /mnt/snapshot

# Remove snapshot after backup
sudo umount /mnt/snapshot
sudo lvremove /dev/myvg/mylv_snapshot
```

Advanced LVM Features

LVM Thin Provisioning

```
# Create thin pool
sudo lvcreate -L 50G --thinpool mythinpool myvg

# Create thin volume
sudo lvcreate -V 100G --thin myvg/mythinpool -n thinlv

# Monitor thin pool usage
sudo lvs -o +data_percent,metadata_percent
```

LVM Caching

```
# Create cache pool (requires SSD)
sudo lvcreate -L 10G -n cachepool myvg /dev/nvme0n1p1

# Convert to cache pool
sudo lvconvert --type cache-pool myvg/cachepool

# Cache a logical volume
sudo lvconvert --type cache --cachepool myvg/cachepool myvg/mylv
```

ZFS Volume Management

Installation on Ubuntu 22.04

```
# Install ZFS
sudo apt install zfsutils-linux

# Load ZFS module
sudo modprobe zfs

# Verify installation
zfs version
```

Creating ZFS Pools

```
# Create simple pool
sudo zpool create mypool /dev/sdb

# Create mirrored pool
sudo zpool create mypool mirror /dev/sdb /dev/sdc

# Create RAIDZ pool (RAID5-like)
sudo zpool create mypool raidz /dev/sdb /dev/sdc /dev/sdd

# Check pool status
sudo zpool status
sudo zpool list
```

Creating ZFS Datasets

```
# Create dataset
sudo zfs create mypool/data

# Create dataset with compression
sudo zfs create -o compression=lz4 mypool/compressed

# Set quota
sudo zfs set quota=10G mypool/data

# List datasets
sudo zfs list
```

ZFS Snapshots and Clones

```
# Create snapshot
sudo zfs snapshot mypool/data@backup-$(date +%Y%m%d)

# List snapshots
sudo zfs list -t snapshot

# Clone snapshot
sudo zfs clone mypool/data@backup-20250715 mypool/data-clone

# Rollback to snapshot
sudo zfs rollback mypool/data@backup-20250715
```

Btrfs Volume Management

Creating Btrfs Volumes

```
# Create single device Btrfs
sudo mkfs.btrfs /dev/sdb

# Create multi-device Btrfs
sudo mkfs.btrfs -d raid1 -m raid1 /dev/sdb /dev/sdc

# Mount Btrfs filesystem
sudo mount /dev/sdb /mnt/btrfs
```

Btrfs Subvolumes

```
# Create subvolume
sudo btrfs subvolume create /mnt/btrfs/subvol1

# List subvolumes
sudo btrfs subvolume list /mnt/btrfs

# Mount subvolume
sudo mount -o subvol=subvol1 /dev/sdb /mnt/subvol1
```

Btrfs Snapshots

```
# Create snapshot
sudo btrfs subvolume snapshot /mnt/btrfs/subvol1 /mnt/btrfs/snapshot1

# Create read-only snapshot
sudo btrfs subvolume snapshot -r /mnt/btrfs/subvol1 /mnt/btrfs/ro-snapshot
```

Volume Management Scripts

LVM Monitoring Script

```
#!/usr/bin/env python3
"""
LVM Volume Monitoring Script for Ubuntu 22.04
"""

import subprocess
import json
import sys

def run_command(cmd):
    """Execute shell command and return output"""
    try:
        result = subprocess.run(cmd, shell=True, capture_output=True, text=True)
        return result.stdout.strip(), result.returncode
    except Exception as e:
        return str(e), 1

def get_pv_info():
    """Get physical volume information"""
    cmd = "sudo pvs --reportformat json"
    output, code = run_command(cmd)
    if code == 0:
        return json.loads(output)
    return {}

def get_vg_info():
    """Get volume group information"""
    cmd = "sudo vgs --reportformat json"
    output, code = run_command(cmd)
    if code == 0:
        return json.loads(output)
    return {}

def get_lv_info():
    """Get logical volume information"""
    cmd = "sudo lvs --reportformat json"
    output, code = run_command(cmd)
    if code == 0:
        return json.loads(output)
    return {}

def check_space_usage():
    """Check for volume groups with high usage"""
    vg_info = get_vg_info()
    alerts = []

    if 'report' in vg_info:
        for vg in vg_info['report'][0]['vg']:
            used_percent = float(vg['vg_used_percent'].rstrip('%'))
            if used_percent > 80:
                alerts.append(f"VG {vg['vg_name']} is {used_percent}% full")

    return alerts

def main():
    print("=== LVM Volume Status ===")

    # Physical Volumes
    pv_info = get_pv_info()
    if 'report' in pv_info:
        print("\nPhysical Volumes:")
        for pv in pv_info['report'][0]['pv']:
            print(f"  {pv['pv_name']}: {pv['pv_size']} ({pv['pv_used']} used)")

    # Volume Groups
```

```
    vg_info = get_vg_info()
    if 'report' in vg_info:
        print("\nVolume Groups:")
        for vg in vg_info['report'][0]['vg']:
            print(f"  {vg['vg_name']}: {vg['vg_size']} ({vg['vg_used_percent']}
used)")

    # Logical Volumes
    lv_info = get_lv_info()
    if 'report' in lv_info:
        print("\nLogical Volumes:")
        for lv in lv_info['report'][0]['lv']:
            print(f"  {lv['lv_name']}: {lv['lv_size']} ({lv['data_percent'] or 'N/A'}%
data)")

    # Check for alerts
    alerts = check_space_usage()
    if alerts:
        print("\n=== ALERTS ===")
        for alert in alerts:
            print(f"WARNING: {alert}")

if __name__ == "__main__":
    main()
```

Automated Backup Script

```
#!/bin/bash
# LVM Snapshot Backup Script

VOLUME_GROUP="myvg"
LOGICAL_VOLUME="mylv"
SNAPSHOT_NAME="${LOGICAL_VOLUME}_backup_$(date +%Y%m%d_%H%M%S)"
BACKUP_DIR="/backup"
SNAPSHOT_SIZE="2G"

# Function to log messages
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S'): $1" | tee -a /var/log/lvm_backup.log
}

# Function to cleanup on exit
cleanup() {
    if [ -n "$SNAPSHOT_CREATED" ]; then
        log_message "Cleaning up snapshot: $SNAPSHOT_NAME"
        sudo umount /mnt/snapshot 2>/dev/null
        sudo lvremove -f /dev/$VOLUME_GROUP/$SNAPSHOT_NAME
    fi
}

# Set trap for cleanup
trap cleanup EXIT

# Create snapshot
log_message "Creating snapshot: $SNAPSHOT_NAME"
if sudo lvcreate -L $SNAPSHOT_SIZE -s -n $SNAPSHOT_NAME
/dev/$VOLUME_GROUP/$LOGICAL_VOLUME; then
    SNAPSHOT_CREATED=1
    log_message "Snapshot created successfully"
else
    log_message "Failed to create snapshot"
    exit 1
fi
```

```
# Mount snapshot
sudo mkdir -p /mnt/snapshot
if sudo mount /dev/$VOLUME_GROUP/$SNAPSHOT_NAME /mnt/snapshot; then
    log_message "Snapshot mounted at /mnt/snapshot"
else
    log_message "Failed to mount snapshot"
    exit 1
fi

# Create backup
BACKUP_FILE="$BACKUP_DIR/backup_${LOGICAL_VOLUME}_$(date +%Y%m%d_%H%M%S).tar.gz"
log_message "Creating backup: $BACKUP_FILE"

if sudo tar -czf "$BACKUP_FILE" -C /mnt/snapshot .; then
    log_message "Backup completed: $BACKUP_FILE"
    log_message "Backup size: $(du -h "$BACKUP_FILE" | cut -f1)"
else
    log_message "Backup failed"
    exit 1
fi
```

Common Tasks and Q&A

Q: How do I extend a volume group with a new disk?

A: Add the new disk as a physical volume and extend the volume group:

```
sudo pvcreate /dev/sdd
sudo vgextend myvg /dev/sdd
```

Q: Can I move data between logical volumes?

A: Yes, use pvmove to migrate data:

```
# Move all data from /dev/sdb to /dev/sdc
sudo pvmove /dev/sdb /dev/sdc
```

Q: How do I remove a disk from LVM?

A: First move data, then remove:

```
sudo pvmove /dev/sdb
sudo vgreduce myvg /dev/sdb
sudo pvremove /dev/sdb
```

Q: What's the difference between thick and thin provisioning?

A: Thick provisioning allocates all space immediately, while thin provisioning allocates space as needed, allowing overcommitment.

Q: How do I monitor thin pool usage?

A: Use lvs with additional columns:

```
sudo lvs -o +data_percent,metadata_percent
```

Best Practices

1. **Regular Monitoring** - Monitor volume group usage - Set up alerts for high usage - Regular snapshot cleanup
2. **Backup Strategy** - Use LVM snapshots for consistent backups - Test restore

procedures - Keep multiple snapshot generations

3. **Performance** - Align volumes with underlying storage - Use appropriate stripe sizes - Consider SSD caching for performance
4. **Security** - Use LUKS encryption for sensitive data - Implement proper access controls - Regular security audits
5. **Disaster Recovery** - Document volume group configurations - Practice recovery procedures - Maintain offsite backups

See Also

- Disk Management
- File Systems
- RAID Systems
- Troubleshooting Guide

Storage Devices

This comprehensive guide covers the various types of storage devices, their characteristics, selection criteria, and management on Ubuntu 22.04 LTS.

Overview of Storage Devices

Storage devices are the physical components that store data persistently. Understanding different types helps in making informed decisions for system design, performance optimization, and capacity planning.

Classification by Technology

Magnetic Storage

: Traditional hard disk drives (HDDs) using magnetic fields

Solid State Storage

: Flash-based storage devices (SSDs) with no moving parts

Optical Storage

: CD, DVD, Blu-ray discs using laser technology

Tape Storage

: Sequential access magnetic tape for long-term archival

Hybrid Storage

: Combination devices (SSHDs) with both magnetic and flash storage

Hard Disk Drives (HDDs)

Technology Overview

HDDs store data on rotating magnetic platters with read/write heads. They offer:

- High capacity at low cost
- Mechanical components subject to wear

- Sequential access performance
- Susceptible to physical shock

Types of HDDs

Desktop HDDs (3.5”)

```
# Check 3.5" HDD information
sudo hdparm -I /dev/sda | grep -E "(Model|Serial|Capacity)"

# Typical characteristics:
# - Capacity: 500GB to 20TB+
# - RPM: 5400, 7200 RPM
# - Interface: SATA 6Gb/s
# - Power: 5-10W
```

Laptop HDDs (2.5”)

```
# Check 2.5" HDD specifications
lsblk -d -o NAME,SIZE,MODEL,TRAN

# Typical characteristics:
# - Capacity: 500GB to 5TB
# - RPM: 5400 RPM (some 7200 RPM)
# - Interface: SATA 6Gb/s
# - Power: 1-3W
```

Enterprise HDDs

```
# Check enterprise HDD features
sudo smartctl -a /dev/sda | grep -E "(Model|Rotation|Power_Cycle)"

# Characteristics:
# - High reliability (MTBF 2M+ hours)
# - 7200-15000 RPM
# - Advanced error correction
# - Vibration resistance
```

HDD Management Commands

```
# Get detailed HDD information
sudo hdparm -I /dev/sda

# Check HDD health with SMART
sudo smartctl -H /dev/sda
sudo smartctl -a /dev/sda

# Test HDD performance
sudo hdparm -t /dev/sda    # Buffered reads
sudo hdparm -T /dev/sda    # Cached reads

# Set HDD parameters
sudo hdparm -S 60 /dev/sda    # Set standby timeout
sudo hdparm -B 254 /dev/sda   # Disable power management
```

Solid State Drives (SSDs)

Technology Overview

SSDs use NAND flash memory with no moving parts, providing:

- Fast random access
- Low latency
- High IOPS capability
- Limited write endurance
- Higher cost per GB

Types of SSDs

SATA SSDs

```
# Identify SATA SSD
lsblk -d -o NAME,SIZE,MODEL,TRAN | grep sata

# Characteristics:
# - Interface: SATA 6Gb/s (600MB/s max)
# - Form factor: 2.5" or mSATA
# - Compatible with HDD infrastructure
```

NVMe SSDs

```
# List NVMe devices
sudo nvme list

# Get NVMe device information
sudo nvme id-ctrl /dev/nvme0n1

# Characteristics:
# - Interface: PCIe (up to 7GB/s)
# - Form factors: M.2, PCIe card, U.2
# - Lower latency than SATA
```

M.2 SSDs

```
# Check M.2 SSD details
sudo lshw -class disk | grep -A 10 "product.*M.2"

# Types:
# - M.2 SATA: Uses SATA protocol
# - M.2 NVMe: Uses NVMe protocol
# - Form factors: 2242, 2260, 2280, 22110
```

SSD Management and Optimization

Enable TRIM Support

```
# Check TRIM support
sudo hdparm -I /dev/sda | grep TRIM

# Enable TRIM (automatic)
sudo systemctl enable fstrim.timer
sudo systemctl start fstrim.timer

# Manual TRIM
sudo fstrim -v /

# Add to fstab for continuous TRIM
# /dev/sda1 / ext4 defaults,discard 0 1
```

SSD Health Monitoring

```
# Check SSD health with smartctl
sudo smartctl -A /dev/sda | grep -E "(Wear|Program|Erase|Health)"

# For NVMe SSDs
sudo nvme smart-log /dev/nvme0n1

# Monitor wear leveling
sudo smartctl -A /dev/sda | grep Wear_Leveling_Count
```

SSD Performance Tuning

```
# Check current I/O scheduler
cat /sys/block/sda/queue/scheduler

# Set optimal scheduler for SSD
echo mq-deadline | sudo tee /sys/block/sda/queue/scheduler

# Disable barriers for better performance (if using UPS)
# mount -o nobarrier /dev/sda1 /mnt
```

NVMe Management

```
# Install NVMe tools
sudo apt install nvme-cli

# List NVMe devices
sudo nvme list

# Get device information
sudo nvme id-ctrl /dev/nvme0n1
sudo nvme id-ns /dev/nvme0n1

# Check NVMe health
sudo nvme smart-log /dev/nvme0n1

# Format NVMe (careful!)
sudo nvme format /dev/nvme0n1 --lbaf=0

# Secure erase
sudo nvme format /dev/nvme0n1 --ses=1
```

Network Attached Storage (NAS)

Types of NAS Devices

Consumer NAS

```
# Typical consumer NAS specs:
# - 1-8 drive bays
# - ARM or low-power x86 CPU
# - 512MB to 8GB RAM
# - Gigabit Ethernet

# Connect to NAS via SMB
sudo mount -t cifs //nas.local/share /mnt/nas \
  -o username=user,password=pass
```

Enterprise NAS

```
# Enterprise NAS features:
# - 8+ drive bays
# - Redundant components
# - 10GbE networking
```

```
# - Advanced management
```

Building DIY NAS with Ubuntu

```
# Install Samba for SMB/CIFS sharing
sudo apt install samba

# Configure Samba share
sudo tee -a /etc/samba/smb.conf << 'EOF'
[storage]
path = /srv/storage
browseable = yes
read only = no
guest ok = no
valid users = @storage
EOF

# Restart Samba
sudo systemctl restart smbd
```

Storage Area Network (SAN)

iSCSI Configuration

iSCSI Target (Server)

```
# Install iSCSI target
sudo apt install tgt

# Create target configuration
sudo tee /etc/tgt/conf.d/storage.conf << 'EOF'
<target iqn.2025-01.com.example:storage>
    backing-store /dev/sdb
    incominguser iscsi-user password123
    outgoinguser iscsi-target secretpass
</target>
EOF

# Restart target service
sudo systemctl restart tgt

# Check target status
sudo tgtadm --mode target --op show
```

iSCSI Initiator (Client)

```
# Install iSCSI initiator
sudo apt install open-iscsi

# Configure authentication
sudo tee -a /etc/iscsi/iscsid.conf << 'EOF'
node.session.auth.authmethod = CHAP
node.session.auth.username = iscsi-user
node.session.auth.password = password123
EOF

# Discover targets
sudo iscsiadm -m discovery -t st -p 192.168.1.100

# Login to target
sudo iscsiadm -m node --login

# Check connected sessions
```

```
sudo iscsiadm -m session
```

Optical Storage

CD/DVD/Blu-ray Management

```
# Install optical drive tools
sudo apt install wodim genisoimage dvd+rw-tools

# Check optical drive capabilities
wodim --devices
sudo hdparm -I /dev/sr0

# Create ISO image
genisoimage -o backup.iso -J -R /home/user/documents

# Burn ISO to disc
wodim -v speed=8 backup.iso

# Mount optical disc
sudo mount /dev/sr0 /mnt/cdrom

# Eject disc
eject /dev/sr0
```

USB Storage Devices

USB Drive Management

```
# List USB storage devices
lsusb | grep -i storage
lsblk | grep -E "(sd[b-z]|sr[0-9])"

# Get USB device information
sudo fdisk -l /dev/sdb
sudo hdparm -I /dev/sdb

# Format USB drive (careful!)
sudo mkfs.ext4 /dev/sdb1
sudo mkfs.vfat -F 32 /dev/sdb1  # For Windows compatibility

# Mount USB drive
sudo mkdir /mnt/usb
sudo mount /dev/sdb1 /mnt/usb

# Safe removal
sudo umount /mnt/usb
sudo eject /dev/sdb
```

USB Performance Optimization

```
# Check USB version and speed
lsusb -t
dmesg | grep -i "usb.*high\|usb.*super"

# Optimize mount options for USB
sudo mount -o async,noatime,nodiratime /dev/sdb1 /mnt/usb

# Disable USB autosuspend for external drives
echo 'SUBSYSTEM=="usb", ATTR{idVendor}=="1234", ATTR{idProduct}=="5678",
ATTR{power/autosuspend}="-1"' | sudo tee /etc/udev/rules.d/50-usb-power.rules
```

Storage Device Monitoring

Health Monitoring Script

```
#!/usr/bin/env python3
"""
Storage Device Health Monitoring Script
"""

import subprocess
import re
import json
import sys
from datetime import datetime

def run_command(cmd):
    """Execute command and return output"""
    try:
        result = subprocess.run(cmd, shell=True, capture_output=True, text=True)
        return result.stdout.strip(), result.returncode
    except Exception as e:
        return str(e), 1

def get_block_devices():
    """Get list of block devices"""
    cmd = "lsblk -J -o NAME,TYPE,SIZE,MODEL,SERIAL"
    output, code = run_command(cmd)
    if code == 0:
        return json.loads(output)['blockdevices']
    return []

def check_smart_health(device):
    """Check SMART health for device"""
    cmd = f"sudo smartctl -H /dev/{device}"
    output, code = run_command(cmd)
    if "PASSED" in output:
        return "HEALTHY"
    elif "FAILED" in output:
        return "FAILED"
    else:
        return "UNKNOWN"

def get_temperature(device):
    """Get device temperature"""
    cmd = f"sudo smartctl -A /dev/{device} | grep -i temperature"
    output, code = run_command(cmd)
    if code == 0 and output:
        temp_match = re.search(r'(\d+)\s*\(.*Celsius', output)
        if temp_match:
            return int(temp_match.group(1))
    return None

def check_nvme_health(device):
    """Check NVMe specific health"""
    cmd = f"sudo nvme smart-log /dev/{device}"
    output, code = run_command(cmd)
    if code == 0:
        health_info = {}
        for line in output.split('\n'):
            if 'temperature' in line.lower():
                temp_match = re.search(r'(\d+)', line)
                if temp_match:
                    health_info['temperature'] = int(temp_match.group(1))
            elif 'percentage_used' in line.lower():
```

```
                percent_match = re.search(r'(\d+)%', line)
                if percent_match:
                    health_info['wear_level'] = int(percent_match.group(1))
        return health_info
    return {}

def main():
    print(f"Storage Device Health Report - {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}")
    print("=" * 70)

    devices = get_block_devices()
    alerts = []

    for device in devices:
        if device['type'] == 'disk':
            name = device['name']
            size = device['size']
            model = device.get('model', 'Unknown')

            print(f"\nDevice: /dev/{name}")
            print(f"Model: {model}")
            print(f"Size: {size}")

            # Check SMART health
            health = check_smart_health(name)
            print(f"Health: {health}")

            if health == "FAILED":
                alerts.append(f"CRITICAL: Device {name} SMART health failed!")

            # Check temperature
            temp = get_temperature(name)
            if temp:
                print(f"Temperature: {temp}°C")
                if temp > 60:
                    alerts.append(f"WARNING: Device {name} temperature high: {temp}
°C")

            # Check NVMe specific data
            if name.startswith('nvme'):
                nvme_health = check_nvme_health(name)
                if 'wear_level' in nvme_health:
                    wear = nvme_health['wear_level']
                    print(f"Wear Level: {wear}%")
                    if wear > 80:
                        alerts.append(f"WARNING: Device {name} wear level high:
{wear}%")

    # Display alerts
    if alerts:
        print("\n" + "=" * 70)
        print("ALERTS:")
        for alert in alerts:
            print(f"  {alert}")
    else:
        print("\nAll devices appear healthy.")

if __name__ == "__main__":
    main()
```

Performance Benchmarking

```
#!/bin/bash
```

```
# Storage Performance Benchmark Script

DEVICE=${1:-/dev/sda}
TEST_FILE="/tmp/storage_test"
BLOCK_SIZES=("4k" "64k" "1M" "16M")

echo "Storage Performance Benchmark for $DEVICE"
echo "========================================="

# Check if device exists
if [ ! -b "$DEVICE" ]; then
    echo "Error: Device $DEVICE not found"
    exit 1
fi

# Install fio if not present
if ! command -v fio &> /dev/null; then
    echo "Installing fio benchmark tool..."
    sudo apt update && sudo apt install -y fio
fi

# Sequential read test
echo -e "\n--- Sequential Read Test ---"
sudo fio --name=seq_read --filename=$DEVICE --rw=read --bs=1M --size=1G --numjobs=1 --
runtime=30 --group_reporting

# Sequential write test
echo -e "\n--- Sequential Write Test ---"
sudo fio --name=seq_write --filename=$DEVICE --rw=write --bs=1M --size=1G --numjobs=1
--runtime=30 --group_reporting

# Random read test
echo -e "\n--- Random Read Test ---"
sudo fio --name=rand_read --filename=$DEVICE --rw=randread --bs=4k --size=1G --
numjobs=4 --runtime=30 --group_reporting

# Random write test
echo -e "\n--- Random Write Test ---"
sudo fio --name=rand_write --filename=$DEVICE --rw=randwrite --bs=4k --size=1G --
numjobs=4 --runtime=30 --group_reporting

# Mixed workload test
echo -e "\n--- Mixed Workload Test (70% read, 30% write) ---"
sudo fio --name=mixed --filename=$DEVICE --rw=randrw --rwmixread=70 --bs=4k --size=1G
--numjobs=4 --runtime=30 --group_reporting
```

Device Selection Guidelines

Performance Considerations

For Operating System: * NVMe SSD for best performance * SATA SSD as budget alternative * Minimum 256GB capacity

For Data Storage: * Large HDDs for bulk storage * SSDs for frequently accessed data * Consider hybrid approach

For Backup: * External HDDs for local backup * Tape drives for archival * Cloud storage for offsite backup

Capacity Planning

```
#!/usr/bin/env python3
```

```
"""
Storage Capacity Planning Calculator
"""

def calculate_storage_needs():
    print("Storage Capacity Planning Calculator")
    print("====================================")

    # Get current usage
    os_size = float(input("Operating system size (GB): ") or "50")
    apps_size = float(input("Applications size (GB): ") or "100")
    user_data = float(input("Current user data (GB): ") or "500")

    # Growth projections
    growth_rate = float(input("Annual growth rate (%): ") or "20") / 100
    years = int(input("Planning period (years): ") or "3")

    # Calculate projected needs
    current_total = os_size + apps_size + user_data
    projected_data = user_data * ((1 + growth_rate) ** years)
    projected_total = os_size + apps_size + projected_data

    # Add overhead for performance and backup
    overhead_factor = 1.3  # 30% overhead
    recommended_capacity = projected_total * overhead_factor

    print(f"\nCapacity Analysis:")
    print(f"Current total usage: {current_total:.1f} GB")
    print(f"Projected data growth: {projected_data:.1f} GB")
    print(f"Projected total usage: {projected_total:.1f} GB")
    print(f"Recommended capacity: {recommended_capacity:.1f} GB")

    # Storage recommendations
    if recommended_capacity < 500:
        print("\nRecommendation: 500GB SSD")
    elif recommended_capacity < 1000:
        print("\nRecommendation: 1TB SSD or 1TB HDD + 256GB SSD")
    elif recommended_capacity < 4000:
        print("\nRecommendation: 4TB HDD + 512GB SSD (hybrid)")
    else:
        print("\nRecommendation: Multiple drive configuration or NAS")

if __name__ == "__main__":
    calculate_storage_needs()
```

Common Issues and Troubleshooting

Q: My SSD performance has degraded over time. What can I do?

A: Check TRIM support, run secure erase, monitor wear leveling:

```
# Check TRIM support
sudo fstrim -v /

# Check SSD health
sudo smartctl -A /dev/sda | grep Wear

# Consider secure erase if needed
sudo hdparm --user-master u --security-set-pass p /dev/sda
sudo hdparm --user-master u --security-erase p /dev/sda
```

Q: How do I recover data from a failing HDD?

A: Use ddrescue for data recovery:

```
# Install ddrescue
sudo apt install gddrescue

# Create image of failing drive
sudo ddrescue -f -n /dev/sda /mnt/backup/drive_image.img /mnt/backup/recovery.log

# Mount image for data recovery
sudo mount -o loop,ro /mnt/backup/drive_image.img /mnt/recovery
```

Q: My USB drive is not recognized. How to troubleshoot?

A: Check USB subsystem and device recognition:

```
# Check USB subsystem
lsusb
dmesg | tail -20

# Check block devices
lsblk

# Try different USB port
# Check filesystem
sudo fsck /dev/sdb1
```

Best Practices

1. **Regular Monitoring** - Monitor SMART health data - Check temperatures regularly - Track performance metrics
2. **Preventive Maintenance** - Enable TRIM for SSDs - Regular defragmentation for HDDs (if needed) - Clean physical connections
3. **Data Protection** - Implement RAID where appropriate - Regular backups - Test restore procedures
4. **Performance Optimization** - Use appropriate I/O schedulers - Optimize filesystem parameters - Consider caching strategies
5. **Lifecycle Management** - Plan for device replacement - Monitor wear indicators - Secure data wiping when disposing

See Also

- Disk Management
- File Systems
- RAID Systems
- Volume Management

Ubuntu 22.04 Setup and Configuration

Ubuntu 22.04 LTS Storage Setup

Ubuntu 22.04 LTS (Jammy Jellyfish) provides excellent storage capabilities out of the box. This section covers the complete setup and configuration process for optimal storage management.

System Requirements and Preparation

Minimum System Requirements

```
Component          | Minimum         | Recommended
-------------------|-----------------|-----------------
RAM                | 4 GB            | 8 GB or more
Storage            | 25 GB           | 100 GB or more
CPU                | 2 GHz dual-core| 2 GHz quad-core
Graphics           | VGA capable     | GPU with driver support
```

Pre-Installation Planning

```
# Check hardware compatibility
lshw -short
lscpu
lsmem
lsblk

# Check UEFI/BIOS mode
[ -d /sys/firmware/efi ] && echo "UEFI" || echo "BIOS"

# Verify secure boot status
mokutil --sb-state
```

Storage-Focused Installation

Partitioning Strategies

Option 1: Simple Layout (Recommended for most users)

```
Device    | Mount Point | Size      | Filesystem | Purpose
----------|-------------|-----------|------------|-----------------
/dev/sda1 | /boot/efi   | 512 MB    | FAT32      | EFI System
/dev/sda2 | /           | 50-100 GB | ext4       | Root filesystem
/dev/sda3 | /home       | Remaining | ext4       | User data
/dev/sda4 | [SWAP]      | 2x RAM    | swap       | Virtual memory
```

Option 2: Advanced Layout (For servers/power users)

```
Device    | Mount Point | Size      | Filesystem | Purpose
----------|-------------|-----------|------------|-----------------
/dev/sda1 | /boot/efi   | 512 MB    | FAT32      | EFI System
/dev/sda2 | /boot       | 1 GB      | ext4       | Boot files
/dev/sda3 | /           | 20 GB     | ext4       | Root filesystem
/dev/sda4 | /var        | 20 GB     | ext4       | Variable data
/dev/sda5 | /tmp        | 5 GB      | ext4       | Temporary files
/dev/sda6 | /home       | Remaining | ext4       | User data
/dev/sda7 | [SWAP]      | 2x RAM    | swap       | Virtual memory
```

Manual Partitioning with fdisk

```
# Start partitioning
sudo fdisk /dev/sda

# Create GPT partition table
Command: g

# Create EFI partition
Command: n
Partition number: 1
First sector: (default)
Last sector: +512M
Command: t
```

```
Partition type: 1 (EFI System)

# Create root partition
Command: n
Partition number: 2
First sector: (default)
Last sector: +50G
# (ext4 by default)

# Create home partition
Command: n
Partition number: 3
First sector: (default)
Last sector: (default - use remaining space)

# Write changes
Command: w
```

LVM Setup for Flexible Storage

```
# Install LVM tools
sudo apt update
sudo apt install lvm2

# Create physical volume
sudo pvcreate /dev/sda3

# Create volume group
sudo vgcreate ubuntu-vg /dev/sda3

# Create logical volumes
sudo lvcreate -L 20G -n root ubuntu-vg
sudo lvcreate -L 10G -n var ubuntu-vg
sudo lvcreate -L 5G -n tmp ubuntu-vg
sudo lvcreate -l 100%FREE -n home ubuntu-vg

# Format logical volumes
sudo mkfs.ext4 /dev/ubuntu-vg/root
sudo mkfs.ext4 /dev/ubuntu-vg/var
sudo mkfs.ext4 /dev/ubuntu-vg/tmp
sudo mkfs.ext4 /dev/ubuntu-vg/home
```

Post-Installation Storage Configuration

Update System and Install Essential Tools

```
# Update package lists and system
sudo apt update && sudo apt upgrade -y

# Install essential storage tools
sudo apt install -y \\
    gdisk \\
    parted \\
    gparted \\
    lvm2 \\
    mdadm \\
    smartmontools \\
    hdparm \\
    iotop \\
    htop \\
    tree \\
    ncdu \\
    rsync \\
```

```
    rclone \\
    fdupes \\
    testdisk \\
    ddrescue \\
    safecopy

# Install filesystem tools
sudo apt install -y \\
    xfsprogs \\
    btrfs-progs \\
    ntfs-3g \\
    exfat-fuse \\
    exfat-utils
```

Configure Storage Monitoring

```
# Enable and configure SMART monitoring
sudo systemctl enable smartmontools
sudo systemctl start smartmontools

# Configure smartd
sudo nano /etc/smartd.conf
# Add: /dev/sda -a -o on -S on -s (S/../../.02|L/../../6/03)

# Enable automatic TRIM for SSDs
sudo systemctl enable fstrim.timer
sudo systemctl start fstrim.timer

# Verify TRIM is working
sudo fstrim -v /
```

Optimize Storage Performance

```
# Check current I/O scheduler
cat /sys/block/sda/queue/scheduler

# Set optimal I/O scheduler for SSDs
echo none | sudo tee /sys/block/sda/queue/scheduler

# Set optimal I/O scheduler for HDDs
echo mq-deadline | sudo tee /sys/block/sda/queue/scheduler

# Make scheduler change permanent
echo 'ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="0",
ATTR{queue/scheduler}="none"' | sudo tee /etc/udev/rules.d/60-ioschedulers.rules
echo 'ACTION=="add|change", KERNEL=="sd[a-z]", ATTR{queue/rotational}=="1",
ATTR{queue/scheduler}="mq-deadline"' | sudo tee -a /etc/udev/rules.d/60-
ioschedulers.rules
```

Storage Security Configuration

Full Disk Encryption with LUKS

```
# Install cryptsetup
sudo apt install cryptsetup

# Encrypt a partition
sudo cryptsetup luksFormat /dev/sdb1

# Open encrypted partition
sudo cryptsetup luksOpen /dev/sdb1 encrypted_drive
```

```
# Create filesystem on encrypted partition
sudo mkfs.ext4 /dev/mapper/encrypted_drive

# Mount encrypted partition
sudo mkdir /mnt/encrypted
sudo mount /dev/mapper/encrypted_drive /mnt/encrypted

# Add to /etc/crypttab for automatic mounting
echo "encrypted_drive /dev/sdb1 none luks" | sudo tee -a /etc/crypttab

# Add to /etc/fstab
echo "/dev/mapper/encrypted_drive /mnt/encrypted ext4 defaults 0 2" | sudo tee -a
/etc/fstab
```

File Permissions and Access Control

```
# Set up secure permissions for user data
sudo chmod 750 /home/username
sudo chown username:username /home/username

# Create shared directories with proper permissions
sudo mkdir /shared
sudo chown root:users /shared
sudo chmod 2775 /shared

# Set up ACLs for fine-grained control
sudo apt install acl

# Example: Give user read/write access to specific directory
sudo setfacl -m u:username:rw /path/to/directory
sudo setfacl -m g:groupname:r /path/to/directory
```

Backup and Recovery Setup

Automated Backup Configuration

```
# Create backup directories
sudo mkdir -p /backup/{daily,weekly,monthly}
sudo mkdir -p /backup/system/{etc,home,var}

# Install backup tools
sudo apt install rsync borgbackup duplicity

# Create backup script
cat > /usr/local/bin/backup_system.sh << 'EOF'
#!/bin/bash

BACKUP_DIR="/backup"
DATE=$(date +%Y%m%d_%H%M%S)
LOG_FILE="/var/log/backup.log"

log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

# Backup system configuration
backup_system() {
    log_message "Starting system backup"

    # Backup /etc
    rsync -av /etc/ "$BACKUP_DIR/system/etc/"

    # Backup user homes
```

```
    rsync -av /home/ "$BACKUP_DIR/system/home/" --exclude=".cache"

    # Backup important /var directories
    rsync -av /var/log/ "$BACKUP_DIR/system/var/log/"
    rsync -av /var/lib/dpkg/ "$BACKUP_DIR/system/var/lib/dpkg/"

    log_message "System backup completed"
}

# Backup user data
backup_data() {
    log_message "Starting data backup"

    for user_dir in /home/*; do
        if [ -d "$user_dir" ]; then
            username=$(basename "$user_dir")
            rsync -av "$user_dir/" "$BACKUP_DIR/daily/$username/" \\
                --exclude=".cache" \\
                --exclude=".thumbnails" \\
                --exclude="Downloads"
        fi
    done

    log_message "Data backup completed"
}

# Create compressed archive
create_archive() {
    log_message "Creating compressed archive"

    tar -czf "$BACKUP_DIR/archive/backup_$DATE.tar.gz" \\
        -C "$BACKUP_DIR" daily system

    # Remove archives older than 30 days
    find "$BACKUP_DIR/archive" -name "*.tar.gz" -mtime +30 -delete

    log_message "Archive created: backup_$DATE.tar.gz"
}

# Main execution
mkdir -p "$BACKUP_DIR/archive"
backup_system
backup_data
create_archive

# Send notification
echo "Backup completed on $(hostname) at $(date)" | \\
    mail -s "Backup Report" admin@localhost 2>/dev/null || true
EOF

chmod +x /usr/local/bin/backup_system.sh

# Schedule backup with cron
echo "0 2 * * * /usr/local/bin/backup_system.sh" | sudo crontab -
```

Storage Monitoring and Alerts

System Health Monitoring

```
# Create monitoring script
cat > /usr/local/bin/storage_monitor.sh << 'EOF'
#!/bin/bash

LOG_FILE="/var/log/storage_monitor.log"
```

```
ALERT_THRESHOLD=85
SMART_LOG="/var/log/smart_check.log"

log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

# Check disk usage
check_disk_usage() {
    log_message "Checking disk usage"

    df -h | grep -vE '^Filesystem|tmpfs|cdrom' | awk '{print $5 " " $1 " " $6}' |
while read usage device mount; do
        usage_percent=$(echo $usage | sed 's/%//g')

        if [ $usage_percent -ge $ALERT_THRESHOLD ]; then
            log_message "ALERT: $device ($mount) is ${usage_percent}% full"
            echo "Disk usage alert: $device ($mount) is ${usage_percent}% full" | \\
                mail -s "Disk Usage Alert - $(hostname)" admin@localhost
        fi
    done
}

# Check SMART status
check_smart_status() {
    log_message "Checking SMART status"

    for device in /dev/sd[a-z]; do
        if [ -b "$device" ]; then
            smart_status=$(smartctl -H "$device" 2>/dev/null | grep "SMART overall-
health" | awk '{print $6}')

            if [ "$smart_status" = "PASSED" ]; then
                log_message "$device: SMART status OK"
            elif [ "$smart_status" = "FAILED" ]; then
                log_message "ALERT: $device SMART status FAILED"
                echo "SMART failure detected on $device" | \\
                    mail -s "SMART Failure Alert - $(hostname)" admin@localhost
            fi

            # Log detailed SMART info
            smartctl -a "$device" >> "$SMART_LOG"
        fi
    done
}

# Check filesystem errors
check_filesystem_errors() {
    log_message "Checking filesystem errors"

    # Check dmesg for filesystem errors
    if dmesg | grep -i "error\|fault\|fail" | grep -i "ext4\|xfs\|btrfs" > /dev/null;
then
        log_message "ALERT: Filesystem errors detected in dmesg"
        dmesg | grep -i "error\|fault\|fail" | grep -i "ext4\|xfs\|btrfs" | \\
            mail -s "Filesystem Error Alert - $(hostname)" admin@localhost
    fi
}

# Check RAID status (if applicable)
check_raid_status() {
    if [ -f /proc/mdstat ]; then
        log_message "Checking RAID status"

        if grep -q "_" /proc/mdstat; then
```

```
            log_message "ALERT: RAID degraded state detected"
            cat /proc/mdstat | mail -s "RAID Alert - $(hostname)" admin@localhost
        fi
    fi
}

# Main execution
check_disk_usage
check_smart_status
check_filesystem_errors
check_raid_status

log_message "Storage monitoring completed"
EOF

chmod +x /usr/local/bin/storage_monitor.sh

# Schedule monitoring
echo "0 */6 * * * /usr/local/bin/storage_monitor.sh" | sudo crontab -
```

Performance Optimization

Kernel Parameter Tuning

```
# Create performance tuning configuration
cat > /etc/sysctl.d/99-storage-performance.conf << 'EOF'
# Storage performance optimizations

# Increase dirty page writeback interval
vm.dirty_writeback_centisecs = 500

# Increase dirty page ratio
vm.dirty_ratio = 20
vm.dirty_background_ratio = 10

# Optimize memory management
vm.swappiness = 10
vm.vfs_cache_pressure = 50

# Increase inotify limits
fs.inotify.max_user_watches = 524288
fs.inotify.max_user_instances = 256

# Optimize network buffer sizes (for network storage)
net.core.rmem_max = 134217728
net.core.wmem_max = 134217728
EOF

# Apply settings
sudo sysctl -p /etc/sysctl.d/99-storage-performance.conf
```

Mount Options Optimization

```
# Optimize /etc/fstab for performance
sudo cp /etc/fstab /etc/fstab.backup

# Example optimized fstab entries
cat >> /etc/fstab << 'EOF'
# Optimized mount options
# SSD mounts with noatime and discard
UUID=your-ssd-uuid /home ext4 defaults,noatime,discard 0 2

# HDD mounts with relatime
```

```
UUID=your-hdd-uuid /data ext4 defaults,relatime 0 2

# Temporary filesystems in RAM
tmpfs /tmp tmpfs defaults,noatime,mode=1777,size=2G 0 0
tmpfs /var/tmp tmpfs defaults,noatime,mode=1777,size=1G 0 0
EOF
```

Frequently Asked Questions

Q: How do I check if my system is using UEFI or BIOS?

A: Use these commands to determine your boot mode:

```
# Check for UEFI
[ -d /sys/firmware/efi ] && echo "UEFI boot" || echo "BIOS boot"

# Check boot mode in more detail
bootctl status

# List EFI variables (UEFI only)
efibootmgr -v
```

Q: What filesystem should I use for different use cases?

A: Recommended filesystems by use case:

```
Use Case                | Recommended Filesystem | Reason
------------------------|------------------------|-----------------
Root partition (/)      | ext4                   | Stable, well-tested
Boot partition (/boot)  | ext4                   | Simple, reliable
Home directories        | ext4                   | Good performance
Large files/media       | XFS                    | Better large file handling
Snapshots needed        | Btrfs                  | Built-in snapshots
Windows compatibility   | NTFS                   | Cross-platform access
USB drives             | exFAT                  | Universal compatibility
```

Q: How do I resize partitions safely in Ubuntu 22.04?

A: Follow these steps for safe partition resizing:

```
# 1. BACKUP YOUR DATA FIRST!

# 2. For ext4 filesystems:
# Unmount the partition
sudo umount /dev/sda2

# Check filesystem
sudo fsck -f /dev/sda2

# Resize partition with parted
sudo parted /dev/sda resizepart 2 100%

# Resize filesystem to match partition
sudo resize2fs /dev/sda2

# 3. For LVM volumes:
# Extend physical volume
sudo pvresize /dev/sda3

# Extend logical volume
sudo lvextend -l +100%FREE /dev/ubuntu-vg/home
```

```
# Resize filesystem
sudo resize2fs /dev/ubuntu-vg/home
```

Q: How do I set up automatic mounting for external drives?

A: Configure automatic mounting:

```
# 1. Get device UUID
sudo blkid /dev/sdb1

# 2. Create mount point
sudo mkdir /mnt/external

# 3. Add to fstab
echo "UUID=your-device-uuid /mnt/external ext4 defaults,user,noauto 0 0" | sudo tee -a
/etc/fstab

# 4. Test mounting
mount /mnt/external

# 5. For auto-mount on insertion (using udev rules)
cat > /etc/udev/rules.d/99-usb-mount.rules << 'EOF'
# Auto-mount USB drives
KERNEL=="sd[a-z][0-9]", SUBSYSTEMS=="usb", ACTION=="add", RUN+="/usr/local/bin/usb-
mount.sh %k"
KERNEL=="sd[a-z][0-9]", SUBSYSTEMS=="usb", ACTION=="remove", RUN+="/usr/local/bin/usb-
unmount.sh %k"
EOF
```

Troubleshooting Common Issues

Boot Issues

```
# Repair GRUB bootloader
sudo grub-install /dev/sda
sudo update-grub

# Boot from live USB and repair
sudo mount /dev/sda2 /mnt
sudo mount /dev/sda1 /mnt/boot/efi
sudo mount --bind /dev /mnt/dev
sudo mount --bind /proc /mnt/proc
sudo mount --bind /sys /mnt/sys
sudo chroot /mnt
grub-install /dev/sda
update-grub
exit
```

Filesystem Corruption

```
# Check and repair ext4 filesystem
sudo fsck.ext4 -f /dev/sda2

# For automatic repair
sudo fsck.ext4 -p /dev/sda2

# For interactive repair
sudo fsck.ext4 /dev/sda2

# Check XFS filesystem
sudo xfs_check /dev/sda2
```

```
# Repair XFS filesystem
sudo xfs_repair /dev/sda2
```

Disk Space Issues

```
# Find large files
sudo find / -type f -size +100M -exec ls -lh {} \\; | awk '{print $9 ": " $5}'

# Clean package cache
sudo apt autoclean
sudo apt autoremove

# Clean logs
sudo journalctl --vacuum-time=7d

# Clean thumbnails and cache
rm -rf ~/.cache/thumbnails/*
rm -rf ~/.cache/*
```

Performance Issues

```
# Check I/O wait
iostat -x 1

# Monitor disk activity
sudo iotop

# Check for filesystem errors
dmesg | grep -i error

# Analyze slow queries (for databases)
sudo apt install sysstat
sar -d 1 10
```

Coding Examples and Scripts

This section provides practical coding examples, scripts, and automation tools for storage management on Ubuntu 22.04.

Storage Monitoring Scripts

System Storage Health Monitor

```
#!/usr/bin/env python3
"""
Storage Health Monitor for Ubuntu 22.04
Monitors disk usage, SMART status, and filesystem health
"""

import subprocess
import json
import sys
import time
from datetime import datetime

class StorageMonitor:
    def __init__(self):
        self.log_file = "/var/log/storage-monitor.log"

    def get_disk_usage(self):
        """Get disk usage for all mounted filesystems"""
        try:
            result = subprocess.run(['df', '-h'], capture_output=True, text=True)
            return result.stdout
        except Exception as e:
            return f"Error getting disk usage: {e}"

    def check_smart_status(self, device):
        """Check SMART status for a device"""
        try:
            result = subprocess.run(['smartctl', '-H', device],
                                    capture_output=True, text=True)
            return "PASSED" in result.stdout
        except Exception as e:
            return False

    def get_mounted_devices(self):
        """Get list of mounted block devices"""
        try:
            result = subprocess.run(['lsblk', '-J'], capture_output=True, text=True)
            data = json.loads(result.stdout)
            devices = []
            for device in data['blockdevices']:
                if device.get('mountpoint'):
                    devices.append(device['name'])
            return devices
        except Exception as e:
            return []
```

```
    def log_status(self, message):
        """Log status message with timestamp"""
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        log_entry = f"[{timestamp}] {message}\n"
        try:
            with open(self.log_file, 'a') as f:
                f.write(log_entry)
        except Exception as e:
            print(f"Error writing to log: {e}")

    def monitor(self):
        """Main monitoring function"""
        print("Storage Health Monitor - Starting...")

        # Check disk usage
        usage = self.get_disk_usage()
        print("Disk Usage:")
        print(usage)

        # Check SMART status for physical drives
        devices = ['/dev/sda', '/dev/sdb', '/dev/nvme0n1']
        for device in devices:
            try:
                smart_ok = self.check_smart_status(device)
                status = "HEALTHY" if smart_ok else "WARNING"
                message = f"SMART status for {device}: {status}"
                print(message)
                self.log_status(message)
            except Exception as e:
                print(f"Could not check {device}: {e}")

        # Alert on high usage
        for line in usage.split('\n')[1:]:
            if line.strip():
                parts = line.split()
                if len(parts) >= 5:
                    usage_pct = parts[4].replace('%', '')
                    if usage_pct.isdigit() and int(usage_pct) > 90:
                        alert = f"HIGH USAGE ALERT: {parts[5]} is {usage_pct}% full"
                        print(alert)
                        self.log_status(alert)

if __name__ == "__main__":
    monitor = StorageMonitor()
    monitor.monitor()
```

Automated Backup Script

```
#!/bin/bash
# Automated Backup Script for Ubuntu 22.04
# Supports incremental backups with rotation

set -euo pipefail

# Configuration
SOURCE_DIR="${1:-/home}"
BACKUP_BASE="/backup"
RETENTION_DAYS=30
LOG_FILE="/var/log/backup.log"
DATE=$(date +%Y%m%d_%H%M%S)
BACKUP_DIR="${BACKUP_BASE}/backup_${DATE}"

# Functions
```

```
log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

check_prerequisites() {
    if [[ $EUID -ne 0 ]]; then
        log_message "ERROR: This script must be run as root"
        exit 1
    fi

    if ! command -v rsync &> /dev/null; then
        log_message "Installing rsync..."
        apt update && apt install -y rsync
    fi

    mkdir -p "$BACKUP_BASE"
}

perform_backup() {
    log_message "Starting backup of $SOURCE_DIR to $BACKUP_DIR"

    # Find most recent backup for incremental
    LATEST_BACKUP=$(find "$BACKUP_BASE" -maxdepth 1 -type d -name "backup_*" | sort |
tail -1)

    if [[ -n "$LATEST_BACKUP" && -d "$LATEST_BACKUP" ]]; then
        log_message "Performing incremental backup from $LATEST_BACKUP"
        rsync -av --link-dest="$LATEST_BACKUP" "$SOURCE_DIR/" "$BACKUP_DIR/"
    else
        log_message "Performing full backup"
        rsync -av "$SOURCE_DIR/" "$BACKUP_DIR/"
    fi

    # Create backup manifest
    find "$BACKUP_DIR" -type f -exec sha256sum {} \; > "${BACKUP_DIR}/manifest.sha256"
    echo "Backup completed: $(date)" > "${BACKUP_DIR}/backup_info.txt"
    echo "Source: $SOURCE_DIR" >> "${BACKUP_DIR}/backup_info.txt"

    log_message "Backup completed successfully"
}

cleanup_old_backups() {
    log_message "Cleaning up backups older than $RETENTION_DAYS days"
    find "$BACKUP_BASE" -maxdepth 1 -type d -name "backup_*" -mtime +$RETENTION_DAYS -
exec rm -rf {} \;
    log_message "Cleanup completed"
}

# Main execution
main() {
    log_message "=== Backup Process Started ==="
    check_prerequisites
    perform_backup
    cleanup_old_backups
    log_message "=== Backup Process Completed ==="
}

# Error handling
trap 'log_message "ERROR: Backup failed on line $LINENO"' ERR

main "$@"
```

RAID Management Tools

RAID Status Checker

```
#!/usr/bin/env python3
"""
RAID Status Checker for Ubuntu 22.04
Monitors software RAID arrays and hardware RAID controllers
"""

import subprocess
import re
import json
from pathlib import Path

class RAIDMonitor:
    def __init__(self):
        self.mdstat_path = Path('/proc/mdstat')

    def check_software_raid(self):
        """Check software RAID status from /proc/mdstat"""
        if not self.mdstat_path.exists():
            return {"status": "no_raid", "message": "No software RAID detected"}

        try:
            with open(self.mdstat_path, 'r') as f:
                content = f.read()

            arrays = []
            lines = content.split('\n')

            for i, line in enumerate(lines):
                if line.startswith('md'):
                    array_info = self.parse_md_line(line, lines[i+1:])
                    arrays.append(array_info)

            return {"status": "active", "arrays": arrays}

        except Exception as e:
            return {"status": "error", "message": str(e)}

    def parse_md_line(self, md_line, following_lines):
        """Parse mdstat line for array information"""
        parts = md_line.split()
        array_name = parts[0]
        array_status = parts[2] if len(parts) > 2 else "unknown"

        # Look for status in following lines
        status_line = ""
        for line in following_lines:
            if line.strip() and not line.startswith('md'):
                status_line = line.strip()
                break

        # Check for rebuild/sync status
        rebuild_match = re.search(r'\[.*\]\s+.*=\s*(\d+\.\d+)%', status_line)

        return {
            "name": array_name,
            "status": array_status,
            "devices": self.extract_devices(md_line),
            "rebuilding": rebuild_match.group(1) if rebuild_match else None,
            "details": status_line
        }

    def extract_devices(self, md_line):
```

```
        """Extract device list from md line"""
        # Simple extraction - could be enhanced
        devices = re.findall(r'[a-z]+\d+\[\d+\]', md_line)
        return [dev.split('[')[0] for dev in devices]

    def check_hardware_raid(self):
        """Check hardware RAID controllers"""
        controllers = []

        # Check for MegaRAID
        try:
            result = subprocess.run(['megacli', '-AdpAllInfo', '-aALL'],
                                    capture_output=True, text=True)
            if result.returncode == 0:
                controllers.append({"type": "MegaRAID", "status": "detected"})
        except FileNotFoundError:
            pass

        # Check for HP Smart Array
        try:
            result = subprocess.run(['hpacucli', 'ctrl', 'all', 'show'],
                                    capture_output=True, text=True)
            if result.returncode == 0:
                controllers.append({"type": "HP Smart Array", "status": "detected"})
        except FileNotFoundError:
            pass

        return controllers

    def get_full_status(self):
        """Get complete RAID status report"""
        report = {
            "timestamp": subprocess.run(['date'], capture_output=True,
text=True).stdout.strip(),
            "software_raid": self.check_software_raid(),
            "hardware_raid": self.check_hardware_raid()
        }

        return report

def main():
    monitor = RAIDMonitor()
    status = monitor.get_full_status()
    print(json.dumps(status, indent=2))

if __name__ == "__main__":
    main()
```

LVM Automation Scripts

Dynamic LV Resize Script

```
#!/bin/bash
# Dynamic Logical Volume Resize Script
# Automatically extends LV when usage exceeds threshold

set -euo pipefail

# Configuration
THRESHOLD=85
EXTEND_SIZE="1G"
LOG_FILE="/var/log/lv-autoresize.log"

log_message() {
```

```
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

check_lv_usage() {
    local lv_path="$1"
    local usage

    usage=$(df "$lv_path" | awk 'NR==2 {print $5}' | sed 's/%//')
    echo "$usage"
}

extend_logical_volume() {
    local lv_path="$1"
    local vg_name="$2"
    local lv_name="$3"

    log_message "Extending $lv_path by $EXTEND_SIZE"

    # Check VG free space
    local free_space
    free_space=$(vgs --noheadings -o vg_free --units g "$vg_name" | awk '{print $1}' |
sed 's/g//')

    if (( $(echo "$free_space < 1" | bc -l) )); then
        log_message "ERROR: Not enough free space in VG $vg_name"
        return 1
    fi

    # Extend LV
    if lvextend -L "+$EXTEND_SIZE" "/dev/$vg_name/$lv_name"; then
        # Resize filesystem
        if resize2fs "/dev/$vg_name/$lv_name"; then
            log_message "Successfully extended $lv_path"
            return 0
        else
            log_message "ERROR: Failed to resize filesystem for $lv_path"
            return 1
        fi
    else
        log_message "ERROR: Failed to extend LV $lv_path"
        return 1
    fi
}

monitor_logical_volumes() {
    # Get all LVs
    while IFS= read -r line; do
        local lv_path vg_name lv_name
        lv_path=$(echo "$line" | awk '{print $1}')
        vg_name=$(echo "$line" | awk '{print $2}')
        lv_name=$(echo "$line" | awk '{print $3}')

        if [[ -n "$lv_path" && "$lv_path" != "LV" ]]; then
            local usage
            usage=$(check_lv_usage "$lv_path")

            if [[ "$usage" -gt "$THRESHOLD" ]]; then
                log_message "WARNING: $lv_path usage is ${usage}%, threshold is
${THRESHOLD}%"
                extend_logical_volume "$lv_path" "$vg_name" "$lv_name"
            else
                log_message "INFO: $lv_path usage is ${usage}% (OK)"
            fi
        fi
    done < <(lvs --noheadings -o lv_path,vg_name,lv_name)
```

```
}

main() {
    if [[ $EUID -ne 0 ]]; then
        echo "This script must be run as root"
        exit 1
    fi

    log_message "=== LV Auto-resize Monitor Started ==="
    monitor_logical_volumes
    log_message "=== LV Auto-resize Monitor Completed ==="
}

main "$@"
```

Network Storage Utilities

NFS Mount Manager

```
#!/usr/bin/env python3
"""
NFS Mount Manager for Ubuntu 22.04
Manages NFS mounts with health checking and auto-recovery
"""

import subprocess
import yaml
import os
import sys
from pathlib import Path

class NFSManager:
    def __init__(self, config_file='/etc/nfs-mounts.yaml'):
        self.config_file = config_file
        self.config = self.load_config()

    def load_config(self):
        """Load NFS mount configuration"""
        default_config = {
            'mounts': [
                {
                    'server': '192.168.1.100',
                    'export': '/export/shared',
                    'mountpoint': '/mnt/nfs-shared',
                    'options': 'rw,sync,hard,intr',
                    'auto_mount': True
                }
            ],
            'timeout': 30,
            'retry_count': 3
        }

        if Path(self.config_file).exists():
            try:
                with open(self.config_file, 'r') as f:
                    return yaml.safe_load(f)
            except Exception as e:
                print(f"Error loading config: {e}")
                return default_config
        else:
            self.save_config(default_config)
            return default_config

    def save_config(self, config):
```

```
    """Save configuration to file"""
    try:
        with open(self.config_file, 'w') as f:
            yaml.dump(config, f, default_flow_style=False)
    except Exception as e:
        print(f"Error saving config: {e}")

def check_nfs_server(self, server):
    """Check if NFS server is reachable"""
    try:
        result = subprocess.run(['ping', '-c', '1', '-W', '5', server],
                                capture_output=True, text=True)
        return result.returncode == 0
    except Exception:
        return False

def is_mounted(self, mountpoint):
    """Check if mountpoint is currently mounted"""
    try:
        result = subprocess.run(['mountpoint', '-q', mountpoint])
        return result.returncode == 0
    except Exception:
        return False

def mount_nfs(self, mount_config):
    """Mount an NFS share"""
    server = mount_config['server']
    export = mount_config['export']
    mountpoint = mount_config['mountpoint']
    options = mount_config.get('options', 'rw,sync')

    print(f"Mounting {server}:{export} -> {mountpoint}")

    # Create mountpoint if it doesn't exist
    Path(mountpoint).mkdir(parents=True, exist_ok=True)

    # Check if already mounted
    if self.is_mounted(mountpoint):
        print(f"  Already mounted: {mountpoint}")
        return True

    # Check server connectivity
    if not self.check_nfs_server(server):
        print(f"  ERROR: Cannot reach NFS server {server}")
        return False

    # Mount the share
    try:
        nfs_source = f"{server}:{export}"
        cmd = ['mount', '-t', 'nfs', '-o', options, nfs_source, mountpoint]
        result = subprocess.run(cmd, capture_output=True, text=True)

        if result.returncode == 0:
            print(f"  Successfully mounted {mountpoint}")
            return True
        else:
            print(f"  ERROR: Mount failed: {result.stderr}")
            return False

    except Exception as e:
        print(f"  ERROR: Exception during mount: {e}")
        return False

def unmount_nfs(self, mountpoint):
    """Unmount an NFS share"""
```

```
    try:
        if self.is_mounted(mountpoint):
            result = subprocess.run(['umount', mountpoint],
                                    capture_output=True, text=True)
            if result.returncode == 0:
                print(f"Successfully unmounted {mountpoint}")
                return True
            else:
                print(f"ERROR: Unmount failed: {result.stderr}")
                return False
        else:
            print(f"{mountpoint} is not mounted")
            return True
    except Exception as e:
        print(f"ERROR: Exception during unmount: {e}")
        return False

def mount_all(self):
    """Mount all configured NFS shares"""
    success_count = 0
    total_count = len(self.config['mounts'])

    for mount_config in self.config['mounts']:
        if mount_config.get('auto_mount', True):
            if self.mount_nfs(mount_config):
                success_count += 1

    print(f"Mounted {success_count}/{total_count} NFS shares")
    return success_count == total_count

def unmount_all(self):
    """Unmount all configured NFS shares"""
    for mount_config in self.config['mounts']:
        self.unmount_nfs(mount_config['mountpoint'])

def health_check(self):
    """Check health of all NFS mounts"""
    print("NFS Mount Health Check")
    print("=" * 50)

    for mount_config in self.config['mounts']:
        server = mount_config['server']
        mountpoint = mount_config['mountpoint']

        print(f"Checking {server}:{mount_config['export']}")

        # Check server
        server_ok = self.check_nfs_server(server)
        print(f"  Server reachable: {'YES' if server_ok else 'NO'}")

        # Check mount
        mounted = self.is_mounted(mountpoint)
        print(f"  Mounted: {'YES' if mounted else 'NO'}")

        # Check accessibility
        if mounted:
            try:
                test_file = Path(mountpoint) / '.nfs_test'
                test_file.touch()
                test_file.unlink()
                print("  Accessible: YES")
            except Exception:
                print("  Accessible: NO")

        print()
```

```
def main():
    if len(sys.argv) < 2:
        print("Usage: nfs-manager.py {mount|unmount|health|mount-all|unmount-all}")
        sys.exit(1)

    if os.geteuid() != 0:
        print("This script must be run as root")
        sys.exit(1)

    manager = NFSManager()
    command = sys.argv[1]

    if command == "mount-all":
        manager.mount_all()
    elif command == "unmount-all":
        manager.unmount_all()
    elif command == "health":
        manager.health_check()
    else:
        print(f"Unknown command: {command}")

if __name__ == "__main__":
    main()
```

File System Utilities

Filesystem Health Checker

```
#!/bin/bash
# Comprehensive Filesystem Health Checker for Ubuntu 22.04

set -euo pipefail

# Configuration
LOG_FILE="/var/log/filesystem-health.log"
REPORT_FILE="/tmp/filesystem-health-report.txt"

log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

check_filesystem_errors() {
    local device="$1"
    local fstype="$2"
    local mountpoint="$3"

    log_message "Checking filesystem errors for $device ($fstype)"

    case "$fstype" in
        ext4|ext3|ext2)
            # Check for ext filesystem errors
            local error_count
            error_count=$(tune2fs -l "$device" 2>/dev/null | grep "Filesystem errors
behavior" || echo "0")

            if dumpe2fs -h "$device" 2>/dev/null | grep -q "has_journal"; then
                log_message "  Journal filesystem detected"
            fi

            # Force check if needed
            local mount_count max_count
            mount_count=$(tune2fs -l "$device" 2>/dev/null | grep "Mount count:" | awk
'{print $3}' || echo "0")
```

```
            max_count=$(tune2fs -l "$device" 2>/dev/null | grep "Maximum mount count:"
| awk '{print $4}' || echo "0")

            if [[ "$max_count" -gt 0 && "$mount_count" -gt "$max_count" ]]; then
                log_message "  WARNING: Mount count ($mount_count) exceeds maximum
($max_count)"
            fi
            ;;

        xfs)
            # Check XFS filesystem
            if xfs_info "$mountpoint" >/dev/null 2>&1; then
                log_message "  XFS filesystem appears healthy"
            else
                log_message "  WARNING: XFS filesystem check failed"
            fi
            ;;

        btrfs)
            # Check Btrfs filesystem
            if btrfs filesystem show "$device" >/dev/null 2>&1; then
                local errors
                errors=$(btrfs device stats "$device" 2>/dev/null | grep -c "err" ||
echo "0")
                log_message "  Btrfs error count: $errors"
            fi
            ;;
    esac
}

check_disk_usage() {
    log_message "Checking disk usage patterns"

    df -h | while IFS= read -r line; do
        if [[ "$line" == *"% /"* ]]; then
            local usage
            usage=$(echo "$line" | awk '{print $5}' | sed 's/%//')
            local mountpoint
            mountpoint=$(echo "$line" | awk '{print $6}')

            if [[ "$usage" -gt 90 ]]; then
                log_message "  CRITICAL: $mountpoint is ${usage}% full"
            elif [[ "$usage" -gt 80 ]]; then
                log_message "  WARNING: $mountpoint is ${usage}% full"
            fi
        fi
    done
}

check_inode_usage() {
    log_message "Checking inode usage"

    df -i | while IFS= read -r line; do
        if [[ "$line" == *"% /"* ]]; then
            local usage
            usage=$(echo "$line" | awk '{print $5}' | sed 's/%//')
            local mountpoint
            mountpoint=$(echo "$line" | awk '{print $6}')

            if [[ "$usage" -gt 90 ]]; then
                log_message "  CRITICAL: $mountpoint inodes ${usage}% used"
            elif [[ "$usage" -gt 80 ]]; then
                log_message "  WARNING: $mountpoint inodes ${usage}% used"
            fi
        fi
```

```
    done
}

generate_report() {
    {
        echo "Filesystem Health Report"
        echo "Generated: $(date)"
        echo "========================================"
        echo

        echo "DISK USAGE:"
        df -h
        echo

        echo "INODE USAGE:"
        df -i
        echo

        echo "MOUNTED FILESYSTEMS:"
        mount | grep -E '^/dev/'
        echo

        echo "FILESYSTEM TYPES:"
        lsblk -f
        echo

        echo "RECENT LOG ENTRIES:"
        tail -20 "$LOG_FILE"

    } > "$REPORT_FILE"

    log_message "Health report generated: $REPORT_FILE"
}

main() {
    if [[ $EUID -ne 0 ]]; then
        echo "This script should be run as root for complete checks"
    fi

    log_message "=== Filesystem Health Check Started ==="

    # Get mounted filesystems
    while IFS= read -r line; do
        if [[ "$line" == /dev/* ]]; then
            local device fstype mountpoint
            device=$(echo "$line" | awk '{print $1}')
            fstype=$(echo "$line" | awk '{print $3}')
            mountpoint=$(echo "$line" | awk '{print $2}')

            check_filesystem_errors "$device" "$fstype" "$mountpoint"
        fi
    done < <(mount | grep -E '^/dev/')

    check_disk_usage
    check_inode_usage
    generate_report

    log_message "=== Filesystem Health Check Completed ==="

    echo "Health check completed. Report available at: $REPORT_FILE"
    echo "Log file: $LOG_FILE"
}

main "$@"
```

Performance Testing Scripts

Storage Benchmark Suite

```
#!/usr/bin/env python3
"""
Storage Performance Benchmark Suite for Ubuntu 22.04
Tests various storage scenarios and generates performance reports
"""

import subprocess
import time
import json
import os
import sys
from pathlib import Path

class StorageBenchmark:
    def __init__(self, test_dir="/tmp/storage_test"):
        self.test_dir = Path(test_dir)
        self.test_dir.mkdir(exist_ok=True)
        self.results = {}

    def run_dd_test(self, test_name, block_size="1M", count=1024):
        """Run DD-based I/O test"""
        test_file = self.test_dir / f"{test_name}.dat"

        # Write test
        write_start = time.time()
        cmd = ['dd', f'if=/dev/zero', f'of={test_file}',
               f'bs={block_size}', f'count={count}', 'conv=fdatasync']

        try:
            result = subprocess.run(cmd, capture_output=True, text=True)
            write_time = time.time() - write_start

            # Parse DD output for speed
            write_speed = self.parse_dd_output(result.stderr)

            # Read test
            read_start = time.time()
            cmd = ['dd', f'if={test_file}', 'of=/dev/null', f'bs={block_size}']
            result = subprocess.run(cmd, capture_output=True, text=True)
            read_time = time.time() - read_start

            read_speed = self.parse_dd_output(result.stderr)

            # Cleanup
            test_file.unlink()

            return {
                'write_time': write_time,
                'read_time': read_time,
                'write_speed': write_speed,
                'read_speed': read_speed,
                'block_size': block_size,
                'data_size': f"{count}{block_size}"
            }

        except Exception as e:
            return {'error': str(e)}

    def parse_dd_output(self, dd_stderr):
        """Parse DD output to extract transfer rate"""
```

```
        import re

        # Look for patterns like "1.0 GB/s" or "500 MB/s"
        pattern = r'(\d+\.?\d*)\s*([KMGT]?B/s)'
        match = re.search(pattern, dd_stderr)

        if match:
            return f"{match.group(1)} {match.group(2)}"
        return "Unknown"

    def run_fio_test(self, test_name, job_config):
        """Run FIO benchmark if available"""
        try:
            # Check if fio is available
            subprocess.run(['which', 'fio'], check=True, capture_output=True)

            # Create FIO job file
            job_file = self.test_dir / f"{test_name}.fio"
            with open(job_file, 'w') as f:
                f.write(job_config)

            # Run FIO
            result = subprocess.run(['fio', str(job_file), '--output-format=json'],
                                    capture_output=True, text=True)

            if result.returncode == 0:
                fio_data = json.loads(result.stdout)
                job_file.unlink()
                return fio_data
            else:
                return {'error': result.stderr}

        except (subprocess.CalledProcessError, FileNotFoundError):
            return {'error': 'FIO not available'}

    def run_sequential_tests(self):
        """Run sequential I/O tests"""
        print("Running sequential I/O tests...")

        # Various block sizes
        block_sizes = ['4K', '64K', '1M', '4M']

        for bs in block_sizes:
            test_name = f"sequential_{bs}"
            print(f"  Testing {bs} blocks...")
            result = self.run_dd_test(test_name, bs, 256 if bs == '4M' else 1024)
            self.results[test_name] = result

    def run_random_tests(self):
        """Run random I/O tests using FIO"""
        print("Running random I/O tests...")

        # Random read test
        random_read_config = """
[random_read]
ioengine=libaio
rw=randread
bs=4k
direct=1
size=100M
numjobs=1
runtime=30
group_reporting
filename={}/random_read.dat
""".format(self.test_dir)
```

```
        result = self.run_fio_test("random_read", random_read_config)
        self.results['random_read'] = result

        # Random write test
        random_write_config = """
[random_write]
ioengine=libaio
rw=randwrite
bs=4k
direct=1
size=100M
numjobs=1
runtime=30
group_reporting
filename={}/random_write.dat
""".format(self.test_dir)

        result = self.run_fio_test("random_write", random_write_config)
        self.results['random_write'] = result

    def generate_report(self):
        """Generate performance report"""
        report_file = self.test_dir / "benchmark_report.json"

        report = {
            'timestamp': time.strftime('%Y-%m-%d %H:%M:%S'),
            'test_directory': str(self.test_dir),
            'system_info': self.get_system_info(),
            'results': self.results
        }

        with open(report_file, 'w') as f:
            json.dump(report, f, indent=2)

        # Generate human-readable summary
        self.print_summary()

        print(f"\nDetailed report saved to: {report_file}")

    def get_system_info(self):
        """Get system information"""
        try:
            # Get CPU info
            cpu_info = subprocess.run(['lscpu'], capture_output=True,
text=True).stdout

            # Get memory info
            mem_info = subprocess.run(['free', '-h'], capture_output=True,
text=True).stdout

            # Get storage info
            storage_info = subprocess.run(['lsblk'], capture_output=True,
text=True).stdout

            return {
                'cpu': cpu_info.split('\n')[:10],  # First 10 lines
                'memory': mem_info,
                'storage': storage_info
            }
        except Exception:
            return {'error': 'Could not gather system info'}

    def print_summary(self):
        """Print benchmark summary"""
```

```
        print("\n" + "="*60)
        print("STORAGE BENCHMARK SUMMARY")
        print("="*60)

        for test_name, result in self.results.items():
            if 'error' not in result:
                print(f"\n{test_name.upper()}:")
                if 'write_speed' in result:
                    print(f"  Write Speed: {result['write_speed']}")
                    print(f"  Read Speed:  {result['read_speed']}")
                elif 'jobs' in result:  # FIO result
                    for job in result['jobs']:
                        if 'read' in job:
                            iops = job['read'].get('iops', 'N/A')
                            bw = job['read'].get('bw', 'N/A')
                            print(f"  Read IOPS:   {iops}")
                            print(f"  Read BW:     {bw} KB/s")
                        if 'write' in job:
                            iops = job['write'].get('iops', 'N/A')
                            bw = job['write'].get('bw', 'N/A')
                            print(f"  Write IOPS:  {iops}")
                            print(f"  Write BW:    {bw} KB/s")
            else:
                print(f"\n{test_name.upper()}: ERROR - {result['error']}")

    def run_all_tests(self):
        """Run complete benchmark suite"""
        print("Starting Storage Benchmark Suite")
        print("This may take several minutes...")

        self.run_sequential_tests()
        self.run_random_tests()
        self.generate_report()

        # Cleanup
        for file in self.test_dir.glob("*.dat"):
            file.unlink()

def main():
    if len(sys.argv) > 1:
        test_dir = sys.argv[1]
    else:
        test_dir = "/tmp/storage_test"

    benchmark = StorageBenchmark(test_dir)
    benchmark.run_all_tests()

if __name__ == "__main__":
    main()
```

Installation and Usage Instructions

Setting Up the Environment

```
# Install required packages
sudo apt update
sudo apt install -y python3-pip smartmontools fio

# Install Python dependencies
pip3 install pyyaml

# Make scripts executable
chmod +x *.py *.sh
```

```
# Create necessary directories
sudo mkdir -p /var/log /backup
```

Script Configuration

Storage Monitor Configuration:

```
# Set up cron job for automated monitoring
echo "0 */6 * * * root /path/to/storage-monitor.py" | sudo tee -a /etc/crontab

# Configure log rotation
sudo tee /etc/logrotate.d/storage-monitor << EOF
/var/log/storage-monitor.log {
    daily
    rotate 30
    compress
    missingok
    create 644 root root
}
EOF
```

NFS Manager Setup:

```
# Install NFS utilities
sudo apt install -y nfs-common

# Create configuration file
sudo tee /etc/nfs-mounts.yaml << EOF
mounts:
  - server: "your-nfs-server.local"
    export: "/export/data"
    mountpoint: "/mnt/nfs-data"
    options: "rw,sync,hard,intr"
    auto_mount: true
timeout: 30
retry_count: 3
EOF
```

Integration Examples

Systemd Service Integration

```
# /etc/systemd/system/storage-monitor.service
[Unit]
Description=Storage Health Monitor
After=multi-user.target

[Service]
Type=oneshot
ExecStart=/usr/local/bin/storage-monitor.py
User=root

[Install]
WantedBy=multi-user.target
```

```
# Enable and start the service
sudo systemctl daemon-reload
sudo systemctl enable storage-monitor.service
sudo systemctl start storage-monitor.service
```

Custom Alerts Integration

```
# Email alert integration
import smtplib
from email.mime.text import MIMEText

def send_alert(subject, message):
    msg = MIMEText(message)
    msg['Subject'] = subject
    msg['From'] = 'storage-monitor@yourserver.com'
    msg['To'] = 'admin@yourserver.com'

    server = smtplib.SMTP('localhost')
    server.send_message(msg)
    server.quit()
```

These scripts provide a comprehensive foundation for storage management automation on Ubuntu 22.04. They can be customized and extended based on specific requirements and integrated into larger system management frameworks.

Troubleshooting Guide

This comprehensive troubleshooting guide covers common storage-related issues on Ubuntu 22.04 and their solutions.

Disk and Partition Issues

Disk Not Detected

Symptoms: - New disk not showing up in lsblk or fdisk -l - System doesn't recognize additional storage

Diagnosis Steps:

```
# Check if disk is physically detected
sudo dmesg | grep -i "sd\|nvme\|ata"

# Check SATA/NVMe connections
lspci | grep -i "sata\|nvme"

# Rescan SCSI bus
echo "- - -" | sudo tee /sys/class/scsi_host/host*/scan

# Check disk health
sudo smartctl -a /dev/sdX
```

Solutions:

1. **Physical Connection Issues:**

   ```
   # Power down and check connections
   sudo shutdown -h now
   # Check SATA/power cables
   # Restart system
   ```

2. **Driver Issues:**

```
# Update system
sudo apt update && sudo apt upgrade

# Install additional drivers if needed
sudo ubuntu-drivers autoinstall
```

3. **BIOS/UEFI Settings:** - Enable AHCI mode - Check storage controller settings - Verify disk is detected in BIOS

Partition Table Corruption

Symptoms: - “Invalid partition table” errors - Disk shows as unallocated space - Boot issues

Diagnosis:

```
# Check partition table
sudo fdisk -l /dev/sdX

# Check for backup GPT
sudo gdisk -l /dev/sdX

# Verify filesystem
sudo file -s /dev/sdX*
```

Recovery Steps:

```
# Backup current state
sudo dd if=/dev/sdX of=/backup/disk-backup.img bs=512 count=2048

# Try to repair GPT
sudo gdisk /dev/sdX
# In gdisk: use 'r' for recovery menu, then 'b' to rebuild MBR

# For MBR partition tables
sudo fdisk /dev/sdX
# Use 'p' to print, 'w' to write if fixable

# Create new partition table if necessary
sudo parted /dev/sdX mklabel gpt
```

Boot Issues

GRUB Boot Loader Problems:

```
# Boot from Ubuntu Live USB
# Mount root filesystem
sudo mount /dev/sdX1 /mnt
sudo mount /dev/sdX2 /mnt/boot  # if separate boot partition

# Bind mount system directories
sudo mount --bind /dev /mnt/dev
sudo mount --bind /proc /mnt/proc
sudo mount --bind /sys /mnt/sys

# Chroot into system
sudo chroot /mnt

# Reinstall GRUB
grub-install /dev/sdX
update-grub

# Exit and reboot
```

```
exit
sudo umount -R /mnt
sudo reboot
```

Missing Boot Partition:

```
# Create new boot partition
sudo parted /dev/sdX mkpart primary ext4 1MiB 512MiB
sudo mkfs.ext4 /dev/sdX1

# Mount and restore boot files
sudo mount /dev/sdX1 /mnt
sudo cp -r /boot/* /mnt/

# Update fstab
echo "UUID=$(blkid -s UUID -o value /dev/sdX1) /boot ext4 defaults 0 2" | sudo tee -a
/etc/fstab
```

Filesystem Issues

Filesystem Corruption

Symptoms: - Read-only filesystem errors - “Input/output error” messages - Files disappearing or becoming inaccessible

Emergency Recovery:

```
# Remount filesystem as read-only
sudo mount -o remount,ro /dev/sdX1

# Backup critical data immediately
sudo dd if=/dev/sdX1 of=/backup/corrupted-fs.img conv=noerror,sync

# Check filesystem
sudo fsck -f /dev/sdX1
```

Ext4 Filesystem Repair:

```
# Unmount filesystem first
sudo umount /dev/sdX1

# Check and repair
sudo e2fsck -f -y /dev/sdX1

# If severely corrupted, try alternative superblock
sudo e2fsck -b 32768 /dev/sdX1

# Force repair if needed
sudo e2fsck -f -y -c /dev/sdX1
```

XFS Filesystem Repair:

```
# XFS repair (filesystem must be unmounted)
sudo umount /dev/sdX1
sudo xfs_repair /dev/sdX1

# If metadata is corrupted
sudo xfs_repair -L /dev/sdX1  # This will clear the log
```

Btrfs Filesystem Repair:

```
# Check Btrfs filesystem
sudo btrfs check /dev/sdX1
```

```
# Repair if needed (dangerous, backup first)
sudo btrfs check --repair /dev/sdX1

# Scrub for data integrity
sudo btrfs scrub start /mnt/btrfs-mount
```

Journal Issues

Ext4 Journal Problems:

```
# Check journal status
sudo tune2fs -l /dev/sdX1 | grep -i journal

# Remove journal (converts to ext2)
sudo tune2fs -O ^has_journal /dev/sdX1

# Add journal back
sudo tune2fs -j /dev/sdX1

# Or recreate journal
sudo e2fsck -f /dev/sdX1
sudo tune2fs -J size=128 /dev/sdX1
```

Full Disk Issues

Disk Space Exhaustion:

```
# Find large files
sudo find / -xdev -type f -size +100M -exec ls -lh {} \; 2>/dev/null

# Find large directories
sudo du -h --max-depth=1 / | sort -hr

# Clean package cache
sudo apt autoremove
sudo apt autoclean

# Clean journal logs
sudo journalctl --vacuum-time=3d

# Clean temporary files
sudo rm -rf /tmp/*
sudo rm -rf /var/tmp/*
```

Inode Exhaustion:

```
# Check inode usage
df -i

# Find directories with many files
sudo find / -xdev -type d -exec sh -c 'echo "$(ls -1 "$1" | wc -l) $1"' _ {} \; | sort
-n | tail -20

# Clean up small files
sudo find /var/log -name "*.log" -type f -size +100M -delete
sudo find /tmp -type f -atime +7 -delete
```

LVM Issues

Physical Volume Problems

PV Not Found:

```
# Scan for PVs
sudo pvscan

# Force rescan
sudo pvscan --cache

# Check PV status
sudo pvdisplay -v

# Restore PV from backup
sudo pvcreate --restorefile /etc/lvm/backup/vg_name --uuid PV_UUID /dev/sdX1
```

Missing PV in VG:

```
# Check VG status
sudo vgdisplay
sudo vgs -o +pv_missing

# Try to activate VG with missing PV
sudo vgchange -ay --partial volume_group_name

# Remove missing PV
sudo vgreduce --removemissing volume_group_name
```

Volume Group Issues

VG Cannot Be Activated:

```
# Check VG metadata
sudo vgck volume_group_name

# Restore VG from backup
sudo vgcfgrestore volume_group_name

# Manual metadata restore
sudo vgcfgrestore -f /etc/lvm/backup/volume_group_name volume_group_name
```

Logical Volume Problems

LV Won’t Mount:

```
# Check LV status
sudo lvdisplay
sudo lvs -a

# Activate LV
sudo lvchange -ay /dev/volume_group/logical_volume

# Check filesystem
sudo fsck /dev/volume_group/logical_volume
```

LV Resize Issues:

```
# Check available space
sudo vgs

# Extend LV
sudo lvextend -L +1G /dev/volume_group/logical_volume

# Resize filesystem
sudo resize2fs /dev/volume_group/logical_volume  # for ext4
```

```
sudo xfs_growfs /mount/point                      # for XFS
```

RAID Issues

Software RAID Problems

Array Degraded:

```
# Check RAID status
cat /proc/mdstat

# Check individual device status
sudo mdadm --detail /dev/md0

# Remove failed device
sudo mdadm --manage /dev/md0 --remove /dev/sdX1

# Add replacement device
sudo mdadm --manage /dev/md0 --add /dev/sdY1

# Monitor rebuild
watch cat /proc/mdstat
```

Array Won't Start:

```
# Try to assemble array
sudo mdadm --assemble /dev/md0 /dev/sd[abc]1

# Force assembly with missing device
sudo mdadm --assemble --force /dev/md0 /dev/sd[ab]1

# Scan and assemble all arrays
sudo mdadm --assemble --scan
```

Superblock Issues:

```
# Check superblock
sudo mdadm --examine /dev/sdX1

# Zero superblock if corrupted
sudo mdadm --zero-superblock /dev/sdX1

# Recreate array (data loss!)
sudo mdadm --create /dev/md0 --level=1 --raid-devices=2 /dev/sda1 /dev/sdb1
```

Hardware RAID Issues

Controller Detection:

```
# Check for RAID controllers
lspci | grep -i raid

# Install management tools
sudo apt install megacli hpacucli  # for MegaRAID and HP Smart Array

# Check controller status
sudo megacli -AdpAllInfo -aALL      # MegaRAID
sudo hpacucli ctrl all show         # HP Smart Array
```

Network Storage Issues

NFS Mount Problems

Mount Fails:

```
# Check NFS server availability
showmount -e nfs-server-ip

# Test network connectivity
ping nfs-server-ip
telnet nfs-server-ip 2049

# Check NFS services
sudo systemctl status nfs-client.target
sudo systemctl start nfs-client.target

# Debug mount
sudo mount -v -t nfs nfs-server:/export /mnt/nfs
```

Stale NFS Handles:

```
# Force unmount
sudo umount -f /mnt/nfs

# Lazy unmount
sudo umount -l /mnt/nfs

# Clear stale handles
sudo mount -o remount /mnt/nfs
```

Samba/CIFS Issues

Authentication Failures:

```
# Test with different authentication
sudo mount -t cifs //server/share /mnt/cifs -o username=user,password=pass,vers=3.0

# Check supported protocols
sudo mount -t cifs //server/share /mnt/cifs -o username=user,vers=1.0

# Use credentials file
echo "username=user" | sudo tee /etc/cifs-credentials
echo "password=pass" | sudo tee -a /etc/cifs-credentials
sudo chmod 600 /etc/cifs-credentials
sudo mount -t cifs //server/share /mnt/cifs -o credentials=/etc/cifs-credentials
```

Performance Issues

Slow Disk Performance

Diagnosis:

```
# Check I/O statistics
iostat -x 1 10

# Monitor disk activity
iotop

# Check for errors
dmesg | grep -i "error\|fail"

# SMART health check
sudo smartctl -a /dev/sdX
```

Optimization:

```
# Check and optimize mount options
sudo mount -o remount,noatime,nodiratime /dev/sdX1

# Adjust I/O scheduler
echo deadline | sudo tee /sys/block/sdX/queue/scheduler

# Optimize for SSD
echo 0 | sudo tee /sys/block/sdX/queue/rotational
sudo fstrim -v /
```

High I/O Wait

Investigation:

```
# Check system load
uptime

# Identify processes causing I/O
iotop -a -o

# Check for swap usage
free -h
swapon --show

# Monitor I/O per process
pidstat -d 1
```

Memory and Cache Issues

Clear Caches:

```
# Drop caches safely
sync
echo 3 | sudo tee /proc/sys/vm/drop_caches

# Adjust swappiness
echo 10 | sudo tee /proc/sys/vm/swappiness
```

Check Memory Usage:

```
# Detailed memory information
cat /proc/meminfo

# Check for memory leaks
ps aux --sort=-%mem | head -10
```

Recovery Procedures

Data Recovery

File Recovery with TestDisk:

```
# Install TestDisk
sudo apt install testdisk

# Run TestDisk for partition recovery
sudo testdisk

# Use PhotoRec for file recovery
sudo photorec
```

DD Rescue for Damaged Disks:

```
# Install ddrescue
sudo apt install gddrescue

# Create image of damaged disk
sudo ddrescue -d -r3 /dev/sdX /path/to/rescue.img /path/to/rescue.log

# Continue rescue operation
sudo ddrescue -d -r3 /dev/sdX /path/to/rescue.img /path/to/rescue.log
```

System Recovery

Boot from Live USB:

```
# Mount root filesystem
sudo mkdir /mnt/system
sudo mount /dev/sdX1 /mnt/system

# Mount other partitions
sudo mount /dev/sdX2 /mnt/system/boot
sudo mount /dev/sdX3 /mnt/system/home

# Chroot into system
sudo mount --bind /dev /mnt/system/dev
sudo mount --bind /proc /mnt/system/proc
sudo mount --bind /sys /mnt/system/sys
sudo chroot /mnt/system
```

Backup Before Recovery:

```
# Create full system backup
sudo dd if=/dev/sdX of=/backup/full-disk.img bs=64K conv=noerror,sync

# Create compressed backup
sudo dd if=/dev/sdX bs=64K conv=noerror,sync | gzip > /backup/disk-backup.img.gz
```

Emergency Procedures

Read-Only Filesystem Recovery

```
# Check why filesystem is read-only
dmesg | tail -50

# Try to remount read-write
sudo mount -o remount,rw /

# If that fails, check filesystem
sudo fsck -f /dev/sdX1

# Force filesystem check on next boot
sudo touch /forcefsck
```

Disk Failure Emergency

```
# Immediate data backup
sudo dd if=/dev/failing_disk of=/backup/emergency.img bs=4096 conv=noerror,sync

# Monitor SMART status
sudo smartctl -t short /dev/sdX
sudo smartctl -a /dev/sdX

# Prepare replacement
sudo fdisk -l /dev/new_disk
```

```
sudo dd if=/backup/emergency.img of=/dev/new_disk bs=4096
```

Preventive Measures

Regular Health Checks

```
# Create monitoring script
cat > /usr/local/bin/storage-health.sh << 'EOF'
#!/bin/bash

# Check disk usage
df -h | awk '$5+0 > 80 {print "WARNING: " $0}'

# Check SMART status
for disk in /dev/sd[a-z]; do
    if [ -e "$disk" ]; then
        smartctl -H "$disk" | grep -q "PASSED" || echo "SMART FAILURE: $disk"
    fi
done

# Check RAID status
if [ -f /proc/mdstat ]; then
    grep -q "_" /proc/mdstat && echo "RAID DEGRADED"
fi
EOF

chmod +x /usr/local/bin/storage-health.sh

# Add to crontab
echo "0 6 * * * root /usr/local/bin/storage-health.sh" >> /etc/crontab
```

Backup Verification

```
# Verify backup integrity
md5sum /backup/important-data.tar.gz > /backup/checksums.md5

# Test restore procedure
tar -tzf /backup/important-data.tar.gz > /dev/null && echo "Backup OK" || echo "Backup
CORRUPTED"
```

Log Monitoring

```
# Monitor for storage errors
tail -f /var/log/syslog | grep -i "error\|fail\|critical"

# Set up logwatch for storage events
sudo apt install logwatch
echo "storage" | sudo tee -a /etc/logwatch/conf/services/storage.conf
```

Advanced Debugging

Kernel Debugging

```
# Enable kernel debugging
echo 1 | sudo tee /proc/sys/kernel/printk

# Check kernel ring buffer
dmesg -T | grep -i "storage\|disk\|ata\|scsi"

# Enable block device debugging
echo 1 | sudo tee /sys/block/sdX/queue/iostats
```

Hardware Diagnostics

```
# Check hardware information
sudo lshw -class disk
sudo lspci -v | grep -A 10 -i storage

# Test memory
sudo apt install memtest86+
# Reboot and select memtest from GRUB menu

# CPU stress test
sudo apt install stress-ng
stress-ng --cpu 4 --timeout 60s
```

This troubleshooting guide provides systematic approaches to diagnose and resolve storage issues on Ubuntu 22.04. Always backup critical data before attempting repairs, and consider professional data recovery services for valuable data on physically damaged drives.

Best Practices and Guidelines

This section outlines industry best practices and proven guidelines for storage management on Ubuntu 22.04.

Storage Planning and Design

Capacity Planning

Growth Assessment:

- Plan for 3-5 years of growth
- Account for data growth rates: 20-40% annually for typical business environments
- Include overhead for snapshots, backups, and temporary files
- Reserve 10-15% free space for optimal performance

Performance Requirements:

```
# Benchmark current workload
iotop -a -o -d 1 | head -20

# Monitor I/O patterns
iostat -x 1 60 | tee io-analysis.log

# Calculate IOPS requirements
# Sequential workloads: 100-500 IOPS
# Random workloads: 1000-10000+ IOPS
# Database workloads: 5000-50000+ IOPS
```

Storage Hierarchy Design:

1. **Hot Data** (frequently accessed) - NVMe SSD for highest performance - Direct attachment or high-speed SAN
2. **Warm Data** (regularly accessed) - SATA SSD or high-performance HDD - Network storage with good connectivity
3. **Cold Data** (archival/backup) - High-capacity HDD - Object storage or tape for long-term retention

Filesystem Selection Guidelines

Ext4 - General Purpose:

```
# Optimal for most workloads
sudo mkfs.ext4 -E stride=32,stripe-width=64 /dev/sdX1

# Mount options for performance
mount -o noatime,nodiratime,data=writeback /dev/sdX1 /mnt
```

Use Cases: Boot partitions, general file storage, databases with simple requirements

XFS - High Performance:

```
# Create with optimal settings
sudo mkfs.xfs -f -s size=4096 -d agcount=8 /dev/sdX1
```

```
# Mount with performance options
mount -o noatime,logbsize=256k,largeio /dev/sdX1 /mnt
```

Use Cases: Large files, high-throughput applications, media streaming

Btrfs - Advanced Features:

```
# Create with metadata redundancy
sudo mkfs.btrfs -m raid1 -d single /dev/sdX1 /dev/sdX2

# Enable compression
mount -o compress=zstd,noatime /dev/sdX1 /mnt
```

Use Cases: Development environments, systems requiring snapshots, data integrity critical applications

ZFS - Enterprise Features:

```
# Install ZFS
sudo apt install zfsutils-linux

# Create pool with redundancy
sudo zpool create -o ashift=12 tank raidz2 /dev/sd[abcd]1

# Create filesystem with compression
sudo zfs create -o compression=lz4 tank/data
```

Use Cases: Enterprise storage, virtualization, backup systems

Partitioning Best Practices

Partition Layout Strategy

Standard Desktop/Server Layout:

```
# Create GPT partition table
sudo parted /dev/sdX mklabel gpt

# UEFI boot partition (512MB)
sudo parted /dev/sdX mkpart primary fat32 1MiB 513MiB
sudo parted /dev/sdX set 1 esp on

# Root partition (20-50GB minimum)
sudo parted /dev/sdX mkpart primary ext4 513MiB 50GiB

# Home partition (remaining space)
sudo parted /dev/sdX mkpart primary ext4 50GiB 100%
```

Server Layout with Separate Partitions:

```
# Boot partition (1GB)
sudo parted /dev/sdX mkpart primary ext4 1MiB 1GiB

# Root partition (20GB)
sudo parted /dev/sdX mkpart primary ext4 1GiB 21GiB

# Var partition (10GB minimum)
sudo parted /dev/sdX mkpart primary ext4 21GiB 31GiB

# Home partition (as needed)
sudo parted /dev/sdX mkpart primary ext4 31GiB 131GiB

# Tmp partition (5GB)
```

```
sudo parted /dev/sdX mkpart primary ext4 131GiB 136GiB
```

LVM-based Layout:

```
# Create LVM partition
sudo parted /dev/sdX mkpart primary 513MiB 100%
sudo parted /dev/sdX set 2 lvm on

# Initialize LVM
sudo pvcreate /dev/sdX2
sudo vgcreate vg0 /dev/sdX2

# Create logical volumes
sudo lvcreate -L 20G -n root vg0
sudo lvcreate -L 10G -n var vg0
sudo lvcreate -L 50G -n home vg0
sudo lvcreate -L 4G -n swap vg0
```

Alignment and Performance

Sector Alignment:

```
# Check disk sector size
sudo fdisk -l /dev/sdX | grep "Sector size"

# For 4K sectors, align to 4096-byte boundaries
sudo parted /dev/sdX mkpart primary 4096s 100%

# For SSDs, align to 1MiB boundaries
sudo parted /dev/sdX mkpart primary 1MiB 100%
```

Security Best Practices

Encryption at Rest

LUKS Full Disk Encryption:

```
# Encrypt partition
sudo cryptsetup luksFormat /dev/sdX1

# Open encrypted device
sudo cryptsetup luksOpen /dev/sdX1 encrypted_root

# Format encrypted device
sudo mkfs.ext4 /dev/mapper/encrypted_root

# Add to crypttab
echo "encrypted_root /dev/sdX1 none luks" | sudo tee -a /etc/crypttab
```

Directory-level Encryption:

```
# Install eCryptfs
sudo apt install ecryptfs-utils

# Encrypt home directory
sudo ecryptfs-migrate-home -u username

# Or encrypt specific directories
sudo mount -t ecryptfs /secure /secure
```

Access Control and Permissions

Principle of Least Privilege:

```
# Set restrictive default permissions
umask 027

# Use groups for shared access
sudo groupadd storage-users
sudo usermod -a -G storage-users username

# Set group ownership and permissions
sudo chgrp -R storage-users /shared/storage
sudo chmod -R 750 /shared/storage
```

File System Extended Attributes:

```
# Enable ACLs on filesystem
sudo mount -o remount,acl /dev/sdX1

# Set Access Control Lists
setfacl -m u:username:rw /secure/file.txt
setfacl -m g:groupname:r /secure/directory

# View ACLs
getfacl /secure/file.txt
```

Immutable Files:

```
# Make file immutable
sudo chattr +i /critical/config.file

# Make file append-only
sudo chattr +a /var/log/audit.log

# View attributes
lsattr /critical/config.file
```

Backup and Recovery Strategies

Backup Strategy Framework

3-2-1 Rule Implementation:

- **3** copies of important data
- **2** different storage media types
- **1** copy stored off-site

Backup Types and Schedule:

```
# Full backup (weekly)
tar -czf /backup/full-$(date +%Y%m%d).tar.gz /home /etc /var/log

# Incremental backup (daily)
rsync -av --link-dest=/backup/last-backup /home/ /backup/incremental-$(date +%Y%m%d)/

# Differential backup (daily)
tar -czf /backup/diff-$(date +%Y%m%d).tar.gz --newer-mtime="1 day ago" /home
```

Automated Backup Script:

```
#!/bin/bash
# /usr/local/bin/backup-manager.sh
```

```
BACKUP_ROOT="/backup"
SOURCE_DIRS="/home /etc /var/log"
RETENTION_DAYS=30
LOG_FILE="/var/log/backup.log"

log_message() {
    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG_FILE"
}

perform_backup() {
    local backup_type="$1"
    local backup_dir="$BACKUP_ROOT/$backup_type-$(date +%Y%m%d_%H%M%S)"

    mkdir -p "$backup_dir"

    case "$backup_type" in
        "full")
            tar -czf "$backup_dir/system.tar.gz" $SOURCE_DIRS
            ;;
        "incremental")
            local last_backup=$(find "$BACKUP_ROOT" -name "incremental-*" | sort |
tail -1)
            rsync -av --link-dest="$last_backup" $SOURCE_DIRS "$backup_dir/"
            ;;
    esac

    # Create checksum
    find "$backup_dir" -type f -exec sha256sum {} \; > "$backup_dir/checksums.sha256"

    log_message "Backup completed: $backup_dir"
}

cleanup_old_backups() {
    find "$BACKUP_ROOT" -type d -mtime +$RETENTION_DAYS -exec rm -rf {} \;
    log_message "Cleaned up backups older than $RETENTION_DAYS days"
}

# Run backup based on day of week
if [ "$(date +%u)" -eq 7 ]; then
    perform_backup "full"
else
    perform_backup "incremental"
fi

cleanup_old_backups
```

Backup Verification:

```
# Verify backup integrity
#!/bin/bash
verify_backup() {
    local backup_dir="$1"

    # Check checksums
    cd "$backup_dir"
    sha256sum -c checksums.sha256

    # Test archive extraction
    if [ -f system.tar.gz ]; then
        tar -tzf system.tar.gz > /dev/null && echo "Archive OK" || echo "Archive
CORRUPTED"
    fi

    # Check file count
```

```
    local file_count=$(find . -type f | wc -l)
    echo "Files in backup: $file_count"
}
```

Recovery Planning

Recovery Time Objective (RTO) and Recovery Point Objective (RPO):

- **Critical Systems:** RTO < 4 hours, RPO < 1 hour
- **Important Systems:** RTO < 24 hours, RPO < 4 hours
- **Standard Systems:** RTO < 72 hours, RPO < 24 hours

Disaster Recovery Procedures:

```
# Create disaster recovery documentation
cat > /etc/disaster-recovery.md << 'EOF'
# Disaster Recovery Procedures

## Critical Information
- Backup Location: /backup and remote://backup-server/
- Recovery Media: Ubuntu 22.04 Live USB
- Key Personnel: admin@company.com, +1-555-0123

## Recovery Steps
1. Boot from recovery media
2. Identify storage devices: `lsblk`
3. Mount backup location
4. Restore from most recent backup
5. Verify system integrity
6. Update disaster recovery log
EOF
```

Performance Optimization

I/O Optimization

I/O Scheduler Tuning:

```
# Check current scheduler
cat /sys/block/sdX/queue/scheduler

# Set optimal scheduler per device type
# For SSDs
echo none | sudo tee /sys/block/nvme0n1/queue/scheduler

# For HDDs
echo mq-deadline | sudo tee /sys/block/sda/queue/scheduler

# Make permanent
echo 'ACTION=="add|change", KERNEL=="sd[a-z]*", ATTR{queue/rotational}=="0",
ATTR{queue/scheduler}="none"' | sudo tee /etc/udev/rules.d/60-ssd-scheduler.rules
```

Read-ahead Optimization:

```
# Check current read-ahead
sudo blockdev --getra /dev/sdX

# Set read-ahead for sequential workloads
sudo blockdev --setra 4096 /dev/sdX  # For large files
sudo blockdev --setra 256 /dev/sdX   # For random access
```

Mount Options for Performance:

```
# High-performance mount options
# /etc/fstab entries:

# For databases (ext4)
/dev/mapper/db-data /var/lib/mysql ext4 noatime,nodiratime,data=writeback,barrier=0 0
2

# For web servers (ext4)
/dev/mapper/web-data /var/www ext4 noatime,nodiratime,data=ordered 0 2

# For temporary files
tmpfs /tmp tmpfs defaults,noatime,size=2G 0 0
```

Memory and Cache Optimization

File System Cache Tuning:

```
# Adjust dirty page writeback
echo 5 | sudo tee /proc/sys/vm/dirty_ratio
echo 2 | sudo tee /proc/sys/vm/dirty_background_ratio

# Reduce swappiness for database servers
echo 1 | sudo tee /proc/sys/vm/swappiness

# Make changes permanent
echo "vm.dirty_ratio = 5" | sudo tee -a /etc/sysctl.conf
echo "vm.dirty_background_ratio = 2" | sudo tee -a /etc/sysctl.conf
echo "vm.swappiness = 1" | sudo tee -a /etc/sysctl.conf
```

Monitoring and Alerting

Proactive Monitoring

Key Metrics to Monitor:

1. **Storage Capacity:** Disk usage, inode usage, growth trends
2. **Performance:** IOPS, throughput, latency, queue depth
3. **Health:** SMART status, error rates, temperature
4. **Availability:** Mount status, filesystem errors, RAID status

Monitoring Script:

```
#!/bin/bash
# /usr/local/bin/storage-monitor.sh

ALERT_EMAIL="admin@company.com"
LOG_FILE="/var/log/storage-monitor.log"

# Thresholds
DISK_USAGE_WARN=80
DISK_USAGE_CRIT=90
INODE_USAGE_WARN=80
LOAD_WARN=2.0

send_alert() {
    local subject="$1"
    local message="$2"
```

```
    echo "$message" | mail -s "$subject" "$ALERT_EMAIL"
    echo "$(date): ALERT - $subject - $message" >> "$LOG_FILE"
}

check_disk_usage() {
    df -h | while read filesystem size used avail percent mountpoint; do
        if [[ "$percent" =~ ([0-9]+)% ]]; then
            usage=${BASH_REMATCH[1]}
            if [ "$usage" -gt $DISK_USAGE_CRIT ]; then
                send_alert "CRITICAL: Disk Usage" "$mountpoint is ${usage}% full"
            elif [ "$usage" -gt $DISK_USAGE_WARN ]; then
                send_alert "WARNING: Disk Usage" "$mountpoint is ${usage}% full"
            fi
        fi
    done
}

check_smart_status() {
    for device in /dev/sd[a-z] /dev/nvme[0-9]n[0-9]; do
        if [ -b "$device" ]; then
            if ! smartctl -H "$device" | grep -q "PASSED"; then
                send_alert "CRITICAL: SMART Failure" "Device $device failed SMART
test"
            fi
        fi
    done
}

check_raid_status() {
    if [ -f /proc/mdstat ]; then
        if grep -q "_" /proc/mdstat; then
            send_alert "WARNING: RAID Degraded" "Software RAID array is degraded"
        fi
    fi
}

# Run checks
check_disk_usage
check_smart_status
check_raid_status
```

Automated Health Reports:

```
#!/bin/bash
# Daily storage health report

REPORT_FILE="/tmp/daily-storage-report.txt"

{
    echo "Daily Storage Health Report - $(date)"
    echo "========================================="
    echo

    echo "DISK USAGE:"
    df -h
    echo

    echo "INODE USAGE:"
    df -i
    echo

    echo "SYSTEM LOAD:"
    uptime
    echo
```

```
    echo "I/O STATISTICS (last hour):"
    sar -d -s $(date -d '1 hour ago' '+%H:%M:%S') | tail -20
    echo

    echo "RAID STATUS:"
    cat /proc/mdstat 2>/dev/null || echo "No software RAID detected"
    echo

    echo "SMART STATUS:"
    for device in /dev/sd[a-z]; do
        if [ -b "$device" ]; then
            echo "$device: $(smartctl -H "$device" | grep overall)"
        fi
    done

} > "$REPORT_FILE"

# Email report
mail -s "Daily Storage Report - $(hostname)" admin@company.com < "$REPORT_FILE"
```

Maintenance Procedures

Regular Maintenance Tasks

Weekly Tasks:

```
#!/bin/bash
# Weekly maintenance script

# SMART tests
for device in /dev/sd[a-z]; do
    if [ -b "$device" ]; then
        smartctl -t long "$device"
    fi
done

# Filesystem checks
find /var/log -name "*.log" -size +100M -exec logrotate -f {} \;

# Clean temporary files
find /tmp -type f -atime +7 -delete
find /var/tmp -type f -atime +30 -delete

# Update package cache
apt update
apt list --upgradable
```

Monthly Tasks:

```
#!/bin/bash
# Monthly maintenance script

# Full system backup verification
/usr/local/bin/verify-backups.sh

# Storage capacity planning
/usr/local/bin/capacity-report.sh

# Security updates
apt upgrade -y

# Performance baseline
/usr/local/bin/performance-benchmark.sh
```

Documentation and Change Management

Documentation Standards

Storage Configuration Documentation:

```
# Create storage inventory
cat > /etc/storage-inventory.yaml << 'EOF'
storage_systems:
  - name: "Primary Storage"
    type: "LVM on RAID1"
    devices: ["/dev/sda", "/dev/sdb"]
    capacity: "2TB"
    filesystem: "ext4"
    mount_points:
      - "/": "50GB"
      - "/home": "1.5TB"
      - "/var": "200GB"
    backup_schedule: "Daily incremental, Weekly full"

  - name: "Database Storage"
    type: "Direct attach NVMe"
    devices: ["/dev/nvme0n1"]
    capacity: "1TB"
    filesystem: "xfs"
    mount_points:
      - "/var/lib/mysql": "800GB"
    backup_schedule: "Hourly snapshots, Daily backup"

network_storage:
  - name: "Archive NFS"
    server: "nas.company.local"
    export: "/export/archive"
    mount_point: "/mnt/archive"
    options: "rw,soft,intr"
EOF
```

Change Management Process:

1. **Document all changes** in /var/log/storage-changes.log
2. **Test changes** in development environment first
3. **Create rollback plan** before implementing
4. **Monitor system** after changes for 24-48 hours

These best practices provide a solid foundation for reliable, secure, and high-performing storage systems on Ubuntu 22.04. Regular review and updates of these practices ensure continued effectiveness as technology and requirements evolve.

Glossary

This glossary provides definitions for storage-related terms and concepts used throughout this documentation.

AHCI

: Advanced Host Controller Interface - A technical standard for SATA host controllers that allows software to communicate with storage devices.

Block Device

A type of device that provides buffered access to hardware devices, allowing data to be read and written in fixed-size blocks (typically 512 bytes or 4096 bytes).

Block Size

The minimum unit of data that a filesystem can allocate. Common block sizes are 4KB, 8KB, and 64KB. Larger block sizes are better for sequential access, while smaller block sizes are more efficient for random access.

BTRFS

B-Tree File System - A modern filesystem for Linux that features snapshots, checksums, compression, and advanced volume management capabilities.

Cache

High-speed storage used to temporarily store frequently accessed data to improve performance. Can be implemented in hardware (disk cache) or software (filesystem cache).

CIFS

Common Internet File System - A network protocol used for sharing files, printers, and other resources between systems, primarily used with Windows networks.

DAS

Direct-Attached Storage - Storage devices directly connected to a single computer without a network connection, such as internal hard drives or external USB drives.

Defragmentation

The process of reorganizing data on a storage device to reduce fragmentation and improve performance. Less relevant for modern filesystems and SSDs.

Device Mapper

A Linux kernel framework that provides a generic way to create virtual layers on top of block devices, used by LVM and device encryption.

Dirty Pages

Memory pages that have been modified but not yet written to storage. The kernel manages when these pages are flushed to disk.

DM-Crypt

Device-mapper crypt target - A Linux kernel module that provides transparent encryption of block devices using the device mapper infrastructure.

ECC

Error-Correcting Code - Technology that detects and corrects data corruption errors in memory and storage systems.

Ext4

: Fourth Extended Filesystem - The default filesystem for many Linux distributions, offering journaling, large file support, and backward compatibility.

Filesystem

: A method of organizing and storing data on storage devices. Examples include ext4, XFS, NTFS, and APFS.

FSTAB

: File Systems Table - A configuration file (/etc/fstab) that defines how disk partitions and storage devices should be mounted at boot time.

GPT

: GUID Partition Table - A modern partitioning scheme that replaces MBR, supporting larger disks and more partitions.

HDD

: Hard Disk Drive - Traditional storage device using spinning magnetic disks and mechanical read/write heads.

Hot Spare

: A standby disk in a RAID array that automatically replaces a failed disk without manual intervention.

I/O

: Input/Output - Operations that transfer data between the computer and storage devices or other external systems.

Inode

: Index node - A data structure that stores metadata about files and directories in Unix-like filesystems, including permissions, timestamps, and pointers to data blocks.

IOPS

: Input/Output Operations Per Second - A performance measurement indicating how many read/write operations a storage device can perform per second.

iSCSI

: Internet Small Computer Systems Interface - A protocol that allows SCSI commands to be sent over IP networks, enabling network-attached storage.

Journal

: A log of filesystem changes used for crash recovery. Journaling filesystems can quickly recover from unexpected shutdowns by replaying the journal.

JBOD

: Just a Bunch of Disks - A storage configuration where multiple disks are used independently without RAID redundancy.

LBA

: Logical Block Addressing - A method of addressing data blocks on storage devices using a linear sequence of block numbers.

Logical Volume

: A virtual storage device created by LVM that can span multiple physical devices and be resized dynamically.

LUKS

: Linux Unified Key Setup - A disk encryption specification and reference implementation for Linux that provides a standard format for encrypted storage.

LUN

: Logical Unit Number - A number used to identify individual devices or logical units within a SCSI target.

LVM

: Logical Volume Manager - A system for managing disk space that provides volume management capabilities including resizing, snapshots, and spanning multiple devices.

MBR

: Master Boot Record - An older partitioning scheme limited to 2TB disks and 4 primary partitions.

mdadm

: Multiple Device Administration - A Linux utility for managing software RAID arrays.

Mount Point

: A directory in the filesystem hierarchy where a storage device or filesystem is attached and made accessible.

NAS

: Network-Attached Storage - File-level storage accessible over a network, typically using protocols like NFS or SMB/CIFS.

NFS

: Network File System - A protocol for sharing files over a network, allowing remote directories to be mounted as if they were local.

NVME

: Non-Volatile Memory Express - A high-performance interface for SSDs that connects directly to the CPU via PCIe lanes.

Partition

: A logical division of a storage device that appears to the operating system as a separate device.

Physical Extent

: The smallest unit of space allocation in LVM, typically 4MB in size.

Physical Volume

: A storage device or partition that has been initialized for use with LVM.

RAID

: Redundant Array of Independent Disks - A technology that combines multiple physical disks into logical units for redundancy, performance, or both.

RAID 0

: Disk striping without redundancy - improves performance but provides no fault tolerance.

RAID 1

: Disk mirroring - provides redundancy by maintaining identical copies of data on multiple disks.

RAID 5

: Distributed parity - requires minimum 3 disks, provides redundancy and improved read performance.

RAID 6

: Double distributed parity - requires minimum 4 disks, can tolerate failure of any two disks.

RAID 10

: Combination of RAID 1 and RAID 0 - provides both redundancy and performance improvements.

RPO

: Recovery Point Objective - The maximum acceptable amount of data loss measured in time (e.g., 1 hour RPO means losing at most 1 hour of data).

RTO

: Recovery Time Objective - The maximum acceptable time to restore service after a failure (e.g., 4 hour RTO means service must be restored within 4 hours).

SAN

: Storage Area Network - A dedicated high-speed network that provides block-level access to storage devices.

SATA

: Serial Advanced Technology Attachment - A computer bus interface for connecting storage devices like hard drives and SSDs.

SCSI

: Small Computer System Interface - A set of standards for connecting and

transferring data between computers and storage devices.

Sector

The smallest addressable unit on a storage device, traditionally 512 bytes but increasingly 4096 bytes (4K sectors).

SMART

Self-Monitoring, Analysis and Reporting Technology - A monitoring system for storage devices that detects and reports various reliability indicators.

Snapshot

A point-in-time copy of a filesystem or volume that captures the state of data at a specific moment, useful for backups and recovery.

SSD

Solid State Drive - Storage device using NAND flash memory with no moving parts, offering better performance and reliability than HDDs.

Swap

Virtual memory space on storage devices used when physical RAM is insufficient. Also called a page file.

Throughput

The amount of data transferred per unit of time, typically measured in MB/s or GB/s.

TRIM

A command that informs SSDs which data blocks are no longer needed, allowing the drive to optimize performance and wear leveling.

UUID

Universally Unique Identifier - A 128-bit identifier used to uniquely identify storage devices and filesystems.

Volume Group

A collection of physical volumes in LVM that provides a pool of storage space for creating logical volumes.

Wear Leveling

A technique used in SSDs to distribute write operations evenly across memory cells to maximize device lifespan.

XFS

A high-performance journaling filesystem originally developed for IRIX, known for excellent scalability and large file support.

ZFS

Zettabyte File System - An advanced filesystem and volume manager that provides features like checksums, compression, snapshots, and data deduplication.

Common Command Abbreviations

blkid

: Block device identifier - Command to display information about block devices and their attributes.

dd

: Data duplicator (historically “disk dump”) - Command for copying and converting raw data between devices.

df

: Disk free - Command to display filesystem disk space usage.

du

: Disk usage - Command to display directory space usage.

fdisk

: Fixed disk - Partition table manipulator for MBR partitions.

fsck

: File system check - Command to check and repair filesystem errors.

gdisk

: GPT fdisk - Partition table manipulator for GPT partitions.

lsblk

: List block devices - Command to display block device information in tree format.

mount

: Mount filesystem - Command to attach filesystems to the directory tree.

parted

: Partition editor - Command-line tool for creating and manipulating partition tables.

rsync

: Remote sync - Command for efficiently synchronizing files and directories.

umount

: Unmount filesystem - Command to detach filesystems from the directory tree.

Storage Unit Definitions

Byte

: The basic unit of digital information, consisting of 8 bits.

KB (Kilobyte)

1,000 bytes (decimal) or 1,024 bytes (binary, more precisely called KiB).

MB (Megabyte)

1,000,000 bytes (decimal) or 1,048,576 bytes (binary, more precisely called MiB).

GB (Gigabyte)

1,000,000,000 bytes (decimal) or 1,073,741,824 bytes (binary, more precisely called GiB).

TB (Terabyte)

1,000,000,000,000 bytes (decimal) or 1,099,511,627,776 bytes (binary, more precisely called TiB).

PB (Petabyte)

1,000,000,000,000,000 bytes (decimal) or 1,125,899,906,842,624 bytes (binary, more precisely called PiB).

Performance Metrics

Bandwidth

The maximum data transfer rate of a storage system, typically measured in MB/s or GB/s.

Latency

The time delay between when a storage operation is requested and when it begins, typically measured in milliseconds.

Queue Depth

The number of pending I/O operations that can be queued for processing by a storage device.

Random I/O

Storage operations that access data at non-sequential locations, typically slower than sequential I/O.

Sequential I/O

Storage operations that access data in a continuous, ordered manner, typically faster than random I/O.

Sustained Transfer Rate

The average data transfer rate over an extended period, excluding burst performance.

RAID Levels Reference

RAID 0

Striping - Data is split across multiple drives for improved performance. No redundancy. Minimum 2 drives.

RAID 1

: Mirroring - Data is duplicated across drives for redundancy. 50% storage efficiency. Minimum 2 drives.

RAID 5

: Distributed parity - Data and parity information spread across all drives. Can lose 1 drive. Minimum 3 drives.

RAID 6

: Double parity - Two parity blocks per stripe. Can lose 2 drives. Minimum 4 drives.

RAID 10

: Mirror of stripes - Combines RAID 1 and RAID 0. High performance and redundancy. Minimum 4 drives.

RAID 50

: Striped RAID 5 - Multiple RAID 5 arrays striped together. Minimum 6 drives.

RAID 60

: Striped RAID 6 - Multiple RAID 6 arrays striped together. Minimum 8 drives.

Network Storage Protocols

AFP

: Apple Filing Protocol - Network protocol for file sharing used primarily by Apple devices.

FTP

: File Transfer Protocol - Standard protocol for transferring files over networks.

HTTP/HTTPS

: Hypertext Transfer Protocol - Web protocol that can also be used for file access via WebDAV.

NFS v3

: Network File System version 3 - Stateless file sharing protocol, widely supported.

NFS v4

: Network File System version 4 - Stateful protocol with improved security and features.

SFTP

: SSH File Transfer Protocol - Secure file transfer protocol that operates over SSH.

SMB/CIFS

Server Message Block / Common Internet File System - File sharing protocol used primarily in Windows environments.

WebDAV

: Web Distributed Authoring and Versioning - Extension to HTTP for collaborative file management.

File System Features

Compression

: Feature that reduces file size by encoding data more efficiently, saving storage space at the cost of CPU overhead.

Copy-on-Write

: Technique where data is not physically copied until it is modified, used in snapshots and some filesystems.

Deduplication

: Process of eliminating duplicate data to reduce storage requirements.

Encryption

: Process of encoding data to prevent unauthorized access, can be implemented at filesystem or block device level.

Journaling

: Technique that logs filesystem changes before committing them, enabling fast recovery after crashes.

Quotas

: Limits placed on storage usage by users or groups to prevent overconsumption of disk space.

Snapshots

: Point-in-time copies of filesystem state that can be used for backups or recovery.

Sparse Files

: Files that contain large blocks of zero bytes that are not actually stored on disk, saving space.

This glossary provides essential terminology for understanding storage concepts in Ubuntu 22.04 and serves as a quick reference for technical terms used throughout the documentation.

Frequently Asked Questions

Storage Fundamentals

Q: What’s the difference between storage and memory?

A: Memory (RAM) is volatile storage that loses data when power is removed, while storage is non-volatile and retains data permanently. Memory provides temporary workspace for active programs, while storage provides long-term data retention.

```
# Check memory usage
free -h

# Check storage usage
df -h

# Show memory and storage together
echo "Memory:" && free -h && echo -e "\\nStorage:" && df -h
```

Q: How do I determine what type of storage device I have?

A: Use these commands to identify your storage devices:

```
# List all block devices
lsblk

# Check if SSD or HDD
cat /sys/block/sda/queue/rotational
# 0 = SSD, 1 = HDD

# Get detailed device information
sudo lshw -class disk

# Check device specifications
sudo hdparm -I /dev/sda | grep -E "(Model|Serial|LBA)"
```

Q: What filesystem should I use for different purposes?

A: Choose based on your specific needs:

```
Purpose                | Recommended Filesystem | Reason
-----------------------|------------------------|-----------------
Ubuntu root partition | ext4                    | Stable, well-tested
Large files (>4GB)    | XFS or ext4            | Better large file support
Snapshots needed       | Btrfs                  | Built-in snapshot capability
Windows compatibility  | NTFS                   | Cross-platform support
USB drives            | exFAT                  | Universal compatibility
Network shares        | ext4 or XFS            | Good network performance
```

File System Management

Q: How do I safely resize a partition without losing data?

A: Follow these steps (ALWAYS backup first):

```
# 1. BACKUP YOUR DATA FIRST!

# 2. Unmount the partition
sudo umount /dev/sda2

# 3. Check filesystem integrity
sudo fsck -f /dev/sda2

# 4. Resize partition (using parted)
sudo parted /dev/sda resizepart 2 100%

# 5. Resize filesystem to match partition
sudo resize2fs /dev/sda2
```

```
# 6. Mount and verify
sudo mount /dev/sda2 /mnt/test
df -h /mnt/test
```

Q: How can I recover deleted files?

A: Recovery depends on how quickly you act and the filesystem type:

```
# Install recovery tools
sudo apt install testdisk photorec extundelete

# For ext4 filesystems (must act quickly)
sudo extundelete /dev/sda1 --restore-file /path/to/deleted/file

# For general file recovery
sudo photorec /dev/sda1

# For partition recovery
sudo testdisk /dev/sda

# Emergency: Stop using the drive immediately
sudo mount -o remount,ro /dev/sda1
```

Q: How do I check and repair filesystem errors?

A: Use appropriate tools for each filesystem type:

```
# For ext4 filesystems
sudo fsck.ext4 -f /dev/sda1     # Force check
sudo fsck.ext4 -p /dev/sda1     # Automatic repair

# For XFS filesystems
sudo xfs_check /dev/sda1        # Check only
sudo xfs_repair /dev/sda1       # Repair

# For Btrfs filesystems
sudo btrfs check /dev/sda1      # Check
sudo btrfs check --repair /dev/sda1  # Repair

# Check all filesystems in fstab
sudo fsck -A -f
```

Disk Management

Q: How do I add a new hard drive to my Ubuntu system?

A: Follow this complete process:

```
# 1. Identify the new drive
sudo fdisk -l
lsblk

# 2. Create partition table
sudo parted /dev/sdb mklabel gpt

# 3. Create partition
sudo parted /dev/sdb mkpart primary ext4 0% 100%

# 4. Format the partition
sudo mkfs.ext4 /dev/sdb1
```

```
# 5. Create mount point
sudo mkdir /mnt/newdrive

# 6. Mount temporarily
sudo mount /dev/sdb1 /mnt/newdrive

# 7. Add to fstab for permanent mounting
echo "UUID=$(sudo blkid -s UUID -o value /dev/sdb1) /mnt/newdrive ext4 defaults 0 2" |
sudo tee -a /etc/fstab

# 8. Test fstab entry
sudo umount /mnt/newdrive
sudo mount -a
```

Q: How can I improve disk performance?

A: Several optimization techniques:

```
# Enable TRIM for SSDs
sudo systemctl enable fstrim.timer

# Optimize I/O scheduler
# For SSDs
echo none | sudo tee /sys/block/sda/queue/scheduler

# For HDDs
echo mq-deadline | sudo tee /sys/block/sda/queue/scheduler

# Optimize mount options
sudo mount -o remount,noatime,discard /dev/sda1

# Disable swap if you have enough RAM
sudo swapoff -a
# Comment out swap in /etc/fstab

# Optimize filesystem
sudo tune2fs -o discard /dev/sda1
```

Q: What should I do if my disk is failing?

A: Take immediate action to protect your data:

```
# 1. Check SMART status
sudo smartctl -a /dev/sda

# 2. If drive is still accessible, backup immediately
sudo dd if=/dev/sda of=/backup/disk_image.img bs=1M status=progress

# 3. Or use ddrescue for damaged drives
sudo apt install gddrescue
sudo ddrescue /dev/sda /backup/disk_image.img /backup/rescue.log

# 4. Stop using the drive immediately
sudo umount /dev/sda1

# 5. Replace the drive and restore from backup
```

RAID Configuration

Q: Which RAID level should I choose?

A: Choose based on your priorities:

```
Priority              | RAID Level | Min Disks | Capacity Loss | Performance
----------------------|------------|-----------|---------------|-------------
Performance Only      | RAID 0     | 2         | None          | Excellent
Basic Redundancy      | RAID 1     | 2         | 50%           | Good Read
Balanced Performance  | RAID 5     | 3         | 1 disk        | Good
High Fault Tolerance  | RAID 6     | 4         | 2 disks       | Moderate
Performance + Safety  | RAID 10    | 4         | 50%           | Excellent
```

Q: How do I replace a failed disk in a RAID array?

A: Steps for replacing a failed RAID disk:

```
# 1. Identify failed disk
cat /proc/mdstat
sudo mdadm --detail /dev/md0

# 2. Mark disk as failed (if not auto-detected)
sudo mdadm --fail /dev/md0 /dev/sdb

# 3. Remove failed disk from array
sudo mdadm --remove /dev/md0 /dev/sdb

# 4. Physically replace the disk
# Power down system if hot-swap not supported

# 5. Add new disk to array
sudo mdadm --add /dev/md0 /dev/sdb

# 6. Monitor rebuild progress
watch cat /proc/mdstat
```

Q: Can I convert between RAID levels?

A: Yes, but with limitations and requirements:

```
# RAID 1 to RAID 5 (requires adding a disk first)
sudo mdadm --add /dev/md0 /dev/sdd
sudo mdadm --grow /dev/md0 --level=5 --raid-devices=3

# RAID 5 to RAID 6 (requires adding a disk)
sudo mdadm --add /dev/md0 /dev/sde
sudo mdadm --grow /dev/md0 --level=6 --raid-devices=4

# Note: ALWAYS backup data before conversion
# Some conversions may not be possible

# Check conversion progress
cat /proc/mdstat
```

Network Storage

Q: Should I use NFS or SMB for file sharing?

A: Choose based on your environment:

```
Scenario                    | Recommended | Reason
----------------------------|-------------|-----------------
Linux-only environment     | NFS         | Better performance, native
Mixed OS (Linux/Windows/Mac)| SMB/CIFS    | Universal compatibility
High-performance computing  | NFS         | Lower overhead
Simple file sharing         | SMB/CIFS    | Easier to configure
Security-critical          | NFS + Kerberos | Better authentication
```

Q: How do I troubleshoot slow network storage?

A: Diagnose and optimize network storage performance:

```
# Check network connectivity
ping storage-server
traceroute storage-server

# Test network bandwidth
iperf3 -c storage-server

# Check mount options
mount | grep -E "(nfs|cifs)"

# Optimize NFS mount options
sudo mount -o remount,rsize=32768,wsize=32768,hard,intr /mnt/nfs

# Monitor network I/O
iftop -i eth0
nethogs

# Check for packet loss
mtr storage-server
```

Q: How do I set up automatic mounting for network drives?

A: Configure persistent network mounts:

```
# For NFS shares
echo "nfs-server:/path/to/share /mnt/nfs nfs defaults,_netdev 0 0" | sudo tee -a
/etc/fstab

# For SMB/CIFS shares with credentials
# Create credentials file
cat > ~/.smbcredentials << EOF
username=myuser
password=mypassword
domain=mydomain
EOF
chmod 600 ~/.smbcredentials

# Add to fstab
echo "//server/share /mnt/smb cifs
credentials=/home/user/.smbcredentials,uid=1000,gid=1000,_netdev 0 0" | sudo tee -a
/etc/fstab

# Test automatic mounting
sudo mount -a
```

Backup and Recovery

Q: What’s the best backup strategy for Ubuntu?

A: Implement the 3-2-1 backup rule:

```
# 3-2-1 Rule: 3 copies, 2 different media types, 1 offsite

# Local backup with rsync
rsync -av --delete /home/user/ /backup/local/

# External drive backup
rsync -av --delete /home/user/ /media/external/backup/
```

```
# Cloud backup with rclone
rclone sync /home/user/ cloud:backup/

# System backup script
cat > /usr/local/bin/backup.sh << 'EOF'
#!/bin/bash
# Comprehensive backup script

# Backup user data
rsync -av /home/ /backup/home/

# Backup system configuration
tar -czf /backup/system-$(date +%Y%m%d).tar.gz /etc /var/lib/dpkg

# Create disk image of root partition
dd if=/dev/sda1 of=/backup/root-$(date +%Y%m%d).img bs=1M
EOF
```

Q: How do I restore from a backup?

A: Restoration process depends on backup type:

```
# Restore from rsync backup
rsync -av /backup/home/ /home/

# Restore from tar archive
cd /
sudo tar -xzf /backup/system-20240101.tar.gz

# Restore from disk image
sudo dd if=/backup/root-20240101.img of=/dev/sda1 bs=1M status=progress

# Restore specific files
sudo tar -xzf /backup/system.tar.gz -C / specific/file/path
```

Troubleshooting

Q: My system won’t boot after storage changes. How do I fix it?

A: Boot from Ubuntu live USB and repair:

```
# Boot from Ubuntu live USB

# Mount root partition
sudo mount /dev/sda2 /mnt

# Mount other partitions
sudo mount /dev/sda1 /mnt/boot/efi  # EFI partition

# Bind mount system directories
sudo mount --bind /dev /mnt/dev
sudo mount --bind /proc /mnt/proc
sudo mount --bind /sys /mnt/sys

# Chroot into system
sudo chroot /mnt

# Repair GRUB
grub-install /dev/sda
update-grub

# Fix fstab if needed
```

```
nano /etc/fstab

# Exit and reboot
exit
sudo reboot
```

Q: How do I diagnose storage performance issues?

A: Use these diagnostic tools:

```
# Monitor I/O in real-time
sudo iotop -o

# Check I/O statistics
iostat -x 1

# Monitor disk usage
watch df -h

# Check for high I/O wait
top  # Look for high %wa (I/O wait)

# Test disk speed
sudo hdparm -Tt /dev/sda

# Check for filesystem errors
dmesg | grep -i error

# Monitor SMART attributes
sudo smartctl -A /dev/sda
```

Q: What should I do if I'm running out of disk space?

A: Free up space systematically:

```
# Find large files
sudo find / -type f -size +100M -exec ls -lh {} \\; 2>/dev/null

# Check directory sizes
sudo du -h --max-depth=1 / | sort -hr

# Clean package cache
sudo apt autoclean
sudo apt autoremove

# Clean system logs
sudo journalctl --vacuum-time=3d

# Clean user caches
rm -rf ~/.cache/*
rm -rf ~/.thumbnails/*

# Find and remove duplicate files
fdupes -r /home/user -d

# Move large files to external storage
mv /home/user/large_files/ /mnt/external/
```

Performance Optimization

Q: How can I optimize Ubuntu for SSD storage?

A: Apply SSD-specific optimizations:

```
# Enable TRIM
sudo systemctl enable fstrim.timer

# Set I/O scheduler to none
echo none | sudo tee /sys/block/sda/queue/scheduler

# Update fstab with SSD-optimized options
# Add noatime,discard to mount options
sudo nano /etc/fstab
# Example: UUID=xxx / ext4 defaults,noatime,discard 0 1

# Reduce swappiness
echo "vm.swappiness=1" | sudo tee -a /etc/sysctl.conf

# Move temporary files to RAM
echo "tmpfs /tmp tmpfs defaults,noatime,mode=1777 0 0" | sudo tee -a /etc/fstab
```

Q: How do I monitor storage health proactively?

A: Set up comprehensive monitoring:

```
# Install monitoring tools
sudo apt install smartmontools sysstat

# Enable SMART monitoring
sudo systemctl enable smartd

# Configure email alerts
sudo nano /etc/smartd.conf
# Add: /dev/sda -a -o on -S on -s (S/../../.02|L/../../6/03) -m admin@example.com

# Create monitoring script
cat > /usr/local/bin/storage_health.sh << 'EOF'
#!/bin/bash

# Check disk usage
df -h | awk 'NR>1 {if($5+0 > 85) print $0}' | mail -s "Disk Usage Alert"
admin@example.com

# Check SMART status
for disk in /dev/sd[a-z]; do
    if [ -b "$disk" ]; then
        if ! smartctl -H "$disk" | grep -q PASSED; then
            echo "SMART failure on $disk" | mail -s "SMART Alert" admin@example.com
        fi
    fi
done
EOF

chmod +x /usr/local/bin/storage_health.sh

# Schedule regular checks
echo "0 6 * * * /usr/local/bin/storage_health.sh" | sudo crontab -
```

Downloads and Resources

This section provides information on how to generate and download the documentation in various formats, along with additional resources for storage management on Ubuntu 22.04.

Documentation Formats

Building the Documentation

This documentation is built using Sphinx and can be generated in multiple formats. Follow these steps to build the documentation locally:

Prerequisites:

```
# Install Sphinx and required dependencies
sudo apt update
sudo apt install python3-pip
pip3 install sphinx sphinx-rtd-theme

# Install additional packages for PDF generation
sudo apt install texlive-latex-recommended texlive-fonts-recommended texlive-latex-
extra

# Install packages for EPUB generation
pip3 install sphinx-epub
```

Building HTML Documentation:

```
# Navigate to the documentation directory
cd /path/to/sphinx/source

# Build HTML documentation
sphinx-build -b html . _build/html

# Open in browser
```

```
firefox _build/html/index.html
```

Building PDF Documentation:

```
# Build LaTeX files first
sphinx-build -b latex . _build/latex

# Generate PDF from LaTeX
cd _build/latex
make

# The PDF will be available as storage-guide.pdf
```

Building EPUB Documentation:

```
# Build EPUB format
sphinx-build -b epub . _build/epub

# The EPUB file will be in _build/epub/
```

Building All Formats:

```
#!/bin/bash
# build-docs.sh - Complete documentation build script

set -e

SOURCE_DIR="."
BUILD_DIR="_build"
PROJECT_NAME="Ubuntu-Storage-Guide"

echo "Building Ubuntu 22.04 Storage Documentation..."

# Clean previous builds
rm -rf "$BUILD_DIR"
mkdir -p "$BUILD_DIR"

# Build HTML
echo "Building HTML documentation..."
sphinx-build -b html "$SOURCE_DIR" "$BUILD_DIR/html"

# Build PDF
echo "Building PDF documentation..."
sphinx-build -b latex "$SOURCE_DIR" "$BUILD_DIR/latex"
cd "$BUILD_DIR/latex"
make
cp *.pdf "../${PROJECT_NAME}.pdf"
cd - > /dev/null

# Build EPUB
echo "Building EPUB documentation..."
sphinx-build -b epub "$SOURCE_DIR" "$BUILD_DIR/epub"
cp "$BUILD_DIR/epub/${PROJECT_NAME}.epub" "$BUILD_DIR/"

# Create downloadable archive
echo "Creating download archive..."
cd "$BUILD_DIR"
tar -czf "${PROJECT_NAME}-docs.tar.gz" html/ *.pdf *.epub
cd - > /dev/null

echo "Documentation build complete!"
echo "Available formats:"
echo "  HTML: $BUILD_DIR/html/index.html"
echo "  PDF:  $BUILD_DIR/${PROJECT_NAME}.pdf"
echo "  EPUB: $BUILD_DIR/${PROJECT_NAME}.epub"
```

```
echo "  Archive: $BUILD_DIR/${PROJECT_NAME}-docs.tar.gz"
```

Download Instructions

Available Download Formats:

Quick Downloads

PDF Format

Complete guide in PDF format
Perfect for offline reading

Download PDF

EPUB Format

E-book format for e-readers
Mobile-friendly reading

Download EPUB

Single HTML

Complete guide in one HTML file
Easy sharing and archiving

Download HTML

Complete Archive

All formats in one ZIP
HTML, PDF, EPUB & Text

Download All

For End Users:

1. **PDF Version:** - Complete documentation in a single PDF file - Best for offline reading and printing - Preserves formatting and includes bookmarks - Professional layout with headers and page numbers

2. **EPUB Version:** - E-book format compatible with most e-readers - Best for mobile devices and e-reader applications - Supports reflowable text and adjustable font sizes - Works with Kindle, Apple Books, Adobe Digital Editions

3. **HTML Version:** - Complete website version with navigation and search - Best for online reading and reference - Includes interactive elements and cross-references - Available as single file or complete website archive

4. **Text Version:** - Plain text format for maximum compatibility - Accessible for screen readers and text processing - Lightweight and universally readable

Quick Download Commands:

```
# Download specific formats using wget or curl

# PDF version
wget /downloads/Ubuntu-Storage-Guide.pdf

# EPUB version
wget /downloads/Ubuntu-Storage-Guide.epub

# Complete archive with all formats
wget /downloads/Ubuntu-Storage-Guide-All-Formats.zip

# Extract complete archive
unzip Ubuntu-Storage-Guide-All-Formats.zip
```

Automation Scripts

Automated Build Script

```
#!/bin/bash
# auto-build-docs.sh - Automated documentation builder with CI/CD integration

set -euo pipefail

# Configuration
PROJECT_ROOT="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
SOURCE_DIR="$PROJECT_ROOT/source"
BUILD_DIR="$PROJECT_ROOT/_build"
OUTPUT_DIR="$PROJECT_ROOT/dist"
VERSION=$(date +%Y.%m.%d)

# Colors for output
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[1;33m'
NC='\033[0m' # No Color

log_info() {
    echo -e "${GREEN}[INFO]${NC} $1"
}

log_warn() {
    echo -e "${YELLOW}[WARN]${NC} $1"
}

log_error() {
    echo -e "${RED}[ERROR]${NC} $1"
}

check_dependencies() {
    log_info "Checking dependencies..."

    local missing_deps=()

    if ! command -v sphinx-build &> /dev/null; then
        missing_deps+=("sphinx")
    fi

    if ! command -v pdflatex &> /dev/null; then
        missing_deps+=("texlive-latex-recommended")
    fi
```

```
    if [ ${#missing_deps[@]} -ne 0 ]; then
        log_error "Missing dependencies: ${missing_deps[*]}"
        log_info "Install with: sudo apt install python3-sphinx texlive-latex-
recommended"
        exit 1
    fi

    log_info "All dependencies satisfied"
}

clean_build() {
    log_info "Cleaning previous builds..."
    rm -rf "$BUILD_DIR" "$OUTPUT_DIR"
    mkdir -p "$BUILD_DIR" "$OUTPUT_DIR"
}

build_html() {
    log_info "Building HTML documentation..."
    if sphinx-build -b html -W --keep-going "$SOURCE_DIR" "$BUILD_DIR/html"; then
        log_info "HTML build successful"
        return 0
    else
        log_error "HTML build failed"
        return 1
    fi
}

build_pdf() {
    log_info "Building PDF documentation..."

    # Build LaTeX first
    if sphinx-build -b latex -W --keep-going "$SOURCE_DIR" "$BUILD_DIR/latex"; then
        cd "$BUILD_DIR/latex"
        if make; then
            log_info "PDF build successful"
            cp *.pdf "$OUTPUT_DIR/Ubuntu-Storage-Guide-${VERSION}.pdf"
            cd - > /dev/null
            return 0
        else
            log_error "PDF compilation failed"
            cd - > /dev/null
            return 1
        fi
    else
        log_error "LaTeX build failed"
        return 1
    fi
}

build_epub() {
    log_info "Building EPUB documentation..."
    if sphinx-build -b epub -W --keep-going "$SOURCE_DIR" "$BUILD_DIR/epub"; then
        cp "$BUILD_DIR/epub"/*.epub "$OUTPUT_DIR/Ubuntu-Storage-Guide-${VERSION}.epub"
        log_info "EPUB build successful"
        return 0
    else
        log_error "EPUB build failed"
        return 1
    fi
}

create_archives() {
    log_info "Creating distribution archives..."

    cd "$BUILD_DIR"
```

```
    # Create HTML archive
    tar -czf "$OUTPUT_DIR/Ubuntu-Storage-Guide-HTML-${VERSION}.tar.gz" html/

    # Create complete archive
    tar -czf "$OUTPUT_DIR/Ubuntu-Storage-Guide-Complete-${VERSION}.tar.gz" \
        html/ latex/ epub/

    cd - > /dev/null

    log_info "Archives created successfully"
}

generate_checksums() {
    log_info "Generating checksums..."
    cd "$OUTPUT_DIR"
    sha256sum * > "checksums-${VERSION}.sha256"
    cd - > /dev/null
}

show_summary() {
    log_info "Build Summary:"
    echo "====================================="
    echo "Version: $VERSION"
    echo "Build Directory: $BUILD_DIR"
    echo "Output Directory: $OUTPUT_DIR"
    echo ""
    echo "Generated Files:"
    ls -lh "$OUTPUT_DIR"
    echo ""
    echo "Total Size: $(du -sh "$OUTPUT_DIR" | cut -f1)"
}

# Main execution
main() {
    log_info "Starting automated documentation build (Version: $VERSION)"

    check_dependencies
    clean_build

    local build_errors=0

    # Build all formats
    build_html || ((build_errors++))
    build_pdf || ((build_errors++))
    build_epub || ((build_errors++))

    if [ $build_errors -eq 0 ]; then
        create_archives
        generate_checksums
        show_summary
        log_info "All builds completed successfully!"
    else
        log_warn "$build_errors build(s) failed, but continuing with available
outputs"
    fi
}

# Handle command line arguments
case "${1:-all}" in
    "html")
        check_dependencies
        clean_build
        build_html
        ;;
```

```
    "pdf")
        check_dependencies
        clean_build
        build_pdf
        ;;
    "epub")
        check_dependencies
        clean_build
        build_epub
        ;;
    "all"|"")
        main
        ;;
    *)
        echo "Usage: $0 [html|pdf|epub|all]"
        exit 1
        ;;
esac
```

Continuous Integration Script

```
# .github/workflows/build-docs.yml - GitHub Actions workflow
name: Build Documentation

on:
  push:
    branches: [ main, develop ]
  pull_request:
    branches: [ main ]

jobs:
  build:
    runs-on: ubuntu-22.04

    steps:
    - uses: actions/checkout@v3

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: '3.10'

    - name: Install dependencies
      run: |
        sudo apt-get update
        sudo apt-get install -y texlive-latex-recommended texlive-fonts-recommended
texlive-latex-extra
        pip install sphinx sphinx-rtd-theme

    - name: Build documentation
      run: |
        chmod +x auto-build-docs.sh
        ./auto-build-docs.sh

    - name: Upload artifacts
      uses: actions/upload-artifact@v3
      with:
        name: documentation
        path: dist/

    - name: Deploy to GitHub Pages
      if: github.ref == 'refs/heads/main'
      uses: peaceiris/actions-gh-pages@v3
      with:
```

```
        github_token: ${{ secrets.GITHUB_TOKEN }}
        publish_dir: _build/html
```

Docker Build Environment

```
# Dockerfile for documentation building
FROM ubuntu:22.04

ENV DEBIAN_FRONTEND=noninteractive

# Install dependencies
RUN apt-get update && apt-get install -y \
    python3 \
    python3-pip \
    texlive-latex-recommended \
    texlive-fonts-recommended \
    texlive-latex-extra \
    make \
    && rm -rf /var/lib/apt/lists/*

# Install Python packages
RUN pip3 install sphinx sphinx-rtd-theme

# Set working directory
WORKDIR /docs

# Copy build script
COPY auto-build-docs.sh /usr/local/bin/
RUN chmod +x /usr/local/bin/auto-build-docs.sh

# Default command
CMD ["/usr/local/bin/auto-build-docs.sh"]

# Docker build and run commands

# Build the Docker image
docker build -t storage-docs-builder .

# Run documentation build
docker run -v $(pwd):/docs storage-docs-builder

# Extract built documentation
docker run -v $(pwd):/docs -v $(pwd)/output:/output \
    storage-docs-builder cp -r /docs/dist/* /output/
```

Additional Resources

Official Documentation Links

Ubuntu Documentation:

- Ubuntu Server Guide - Official Ubuntu Server documentation
- Ubuntu Storage Guide - Community filesystem documentation
- Ubuntu LVM Guide - Logical Volume Management

Filesystem Documentation:

- Ext4 Documentation - Kernel documentation for ext4
- XFS Documentation - XFS filesystem wiki

- Btrfs Documentation - Btrfs filesystem wiki
- ZFS on Linux - OpenZFS documentation

RAID and Storage:

- Linux RAID Wiki - Comprehensive RAID documentation
- LVM HOWTO - Linux Documentation Project LVM guide
- Storage Performance - Kernel block layer documentation

Useful Tools and Utilities

System Information:

```
# Essential storage tools installation
sudo apt install -y \
    lvm2 \
    mdadm \
    smartmontools \
    parted \
    gparted \
    hdparm \
    iotop \
    sysstat \
    tree \
    ncdu \
    htop
```

Monitoring Tools:

```
# Advanced monitoring tools
sudo apt install -y \
    nagios-plugins-basic \
    zabbix-agent \
    collectd \
    grafana \
    prometheus-node-exporter
```

Backup Tools:

```
# Backup and recovery tools
sudo apt install -y \
    rsync \
    rdiff-backup \
    duplicity \
    borgbackup \
    testdisk \
    photorec \
    ddrescue
```

Scripts and Templates

Makefile for Documentation:

```
# Makefile for Ubuntu Storage Documentation

SOURCEDIR = source
BUILDDIR = _build
OUTPUTDIR = dist
VERSION = $(shell date +%Y.%m.%d)
PROJECT = Ubuntu-Storage-Guide
```

```
.PHONY: help clean html pdf epub all archive

help:
	@echo "Ubuntu Storage Documentation Build System"
	@echo ""
	@echo "Available targets:"
	@echo "  html     Build HTML documentation"
	@echo "  pdf      Build PDF documentation"
	@echo "  epub     Build EPUB documentation"
	@echo "  all      Build all formats"
	@echo "  archive  Create distribution archives"
	@echo "  clean    Remove build artifacts"
	@echo "  install  Install dependencies"

clean:
	rm -rf $(BUILDDIR) $(OUTPUTDIR)
	@echo "Build artifacts cleaned"

install:
	sudo apt update
	sudo apt install -y python3-sphinx texlive-latex-recommended
	pip3 install sphinx-rtd-theme
	@echo "Dependencies installed"

html:
	mkdir -p $(BUILDDIR)
	sphinx-build -b html $(SOURCEDIR) $(BUILDDIR)/html
	@echo "HTML documentation built in $(BUILDDIR)/html"

pdf:
	mkdir -p $(BUILDDIR) $(OUTPUTDIR)
	sphinx-build -b latex $(SOURCEDIR) $(BUILDDIR)/latex
	cd $(BUILDDIR)/latex && make
	cp $(BUILDDIR)/latex/*.pdf $(OUTPUTDIR)/$(PROJECT)-$(VERSION).pdf
	@echo "PDF documentation built: $(OUTPUTDIR)/$(PROJECT)-$(VERSION).pdf"

epub:
	mkdir -p $(BUILDDIR) $(OUTPUTDIR)
	sphinx-build -b epub $(SOURCEDIR) $(BUILDDIR)/epub
	cp $(BUILDDIR)/epub/*.epub $(OUTPUTDIR)/$(PROJECT)-$(VERSION).epub
	@echo "EPUB documentation built: $(OUTPUTDIR)/$(PROJECT)-$(VERSION).epub"

all: html pdf epub
	@echo "All documentation formats built successfully"

archive: all
	mkdir -p $(OUTPUTDIR)
	cd $(BUILDDIR) && tar -czf ../$(OUTPUTDIR)/$(PROJECT)-HTML-$(VERSION).tar.gz
html/
	cd $(OUTPUTDIR) && tar -czf $(PROJECT)-Complete-$(VERSION).tar.gz *.pdf *.epub
*.tar.gz
	cd $(OUTPUTDIR) && sha256sum * > checksums-$(VERSION).sha256
	@echo "Distribution archives created in $(OUTPUTDIR)"
```

Community Resources

Forums and Communities:

- Ubuntu Forums - Storage Section
- Ask Ubuntu - Storage Questions
- Reddit r/Ubuntu

- Ubuntu Discourse

Professional Resources:

- Linux Professional Institute (LPI) - Linux certification
- Red Hat Training - Enterprise Linux training
- SUSE Training - SUSE Linux Enterprise training

Books and Publications:

- “Linux System Administration” by Tom Adelstein
- “UNIX and Linux System Administration Handbook” by Evi Nemeth
- “Linux Storage Best Practices” by various authors
- “ZFS Administration Guide” by Oracle

Getting Help and Support

Issue Reporting

If you find errors in this documentation or have suggestions for improvement:

1. **Documentation Issues:** - Create an issue on the project repository - Include specific page/section references - Provide detailed description of the problem
2. **Technical Issues:** - Check the troubleshooting section first - Search existing forums and documentation - Provide system information and error messages
3. **Feature Requests:** - Suggest new topics or sections - Provide use cases and justification - Consider contributing content yourself

Contributing to Documentation

This documentation is open source and welcomes contributions:

```
# Fork the repository
git clone https://github.com/username/ubuntu-storage-docs.git
cd ubuntu-storage-docs

# Create a new branch
git checkout -b feature/new-content

# Make your changes
# Edit .rst files in the source/ directory

# Build and test
make html

# Commit and push
git add .
git commit -m "Add new storage topic: XYZ"
git push origin feature/new-content

# Create a pull request
```

Contribution Guidelines:

- Follow the existing documentation style
- Include practical examples and code snippets
- Test all commands on Ubuntu 22.04
- Update the table of contents if adding new sections
- Include relevant cross-references

Contact Information

For questions, suggestions, or support:

- **Email:** docs@storage-guide.org
- **IRC:** #ubuntu-storage on Libera.Chat
- **Matrix:** #ubuntu-storage:matrix.org
- **GitHub:** https://github.com/ubuntu-storage-docs

This comprehensive documentation package provides everything needed to understand and manage storage systems on Ubuntu 22.04, available in multiple formats for different use cases and preferences.

Quick Start Guide

This documentation is designed for:

- System administrators managing storage on Ubuntu 22.04 LTS
- Developers working with storage APIs and file systems
- Students learning about storage technologies
- IT professionals preparing for storage-related certifications

Download Options:

- HTML Documentation (browsable online)
- PDF Version (for offline reading)
- EPUB Format (for e-readers)

Navigation Tips

Use the sidebar to navigate through different sections. Each chapter includes:

- Theoretical concepts and explanations
- Practical examples with Ubuntu 22.04 commands
- Code samples and scripts
- Common questions and answers
- Troubleshooting guides

Indices and tables

- Index
- Module Index
- Search Page
