100 Bash Scripts by CybeCloud

Bash Problem 1

Write a script to check disk space and send alerts if usage exceeds 80%

```
#!/bin/bash

THRESHOLD=80
LOGFILE="disk_usage.log"

echo "Checking disk usage..." > $LOGFILE
while read -r line; do
    usage=$(echo $line | awk '{print $5}' | sed 's/%//')
    partition=$(echo $line | awk '{print $1}')
    if [ "$usage" -ge "$THRESHOLD" ]; then
        echo "Alert: Partition $partition is at $usage% usage!" | tee -a $LOGFILE
    fi
done < <(df -h | grep '^/dev/')

echo "Check completed. Logs saved in $LOGFILE."
```

Bash Problem 2

Automate log rotation and compression.

```
#!/bin/bash

LOG_DIR="/var/log/myapp"
ARCHIVE_DIR="/var/log/myapp/archive"
DATE=$(date +%F)

mkdir -p $ARCHIVE_DIR

for log in $LOG_DIR/*.log; do
    filename=$(basename "$log")
    gzip -c "$log" > "$ARCHIVE_DIR/${filename%.log}-$DATE.log.gz"
    > "$log"  # Clear the log
    echo "Archived and cleared: $log"
done

echo "Log rotation completed. Archived logs are in $ARCHIVE_DIR."
```

Bash Problem 3

Parse a log file to count error occurrences.

```
#!/bin/bash

LOG_FILE="/var/log/syslog"
ERROR_COUNT=$(grep -c "ERROR" $LOG_FILE)

echo "Total ERROR occurrences in $LOG_FILE: $ERROR_COUNT"
```

Bash Problem 4

Automate service restarts if they fail.

```
#!/bin/bash

SERVICE="apache2"
if ! systemctl is-active --quiet $SERVICE; then
    echo "$SERVICE is not running. Restarting..."
    systemctl restart $SERVICE
    echo "$SERVICE restarted."
else
    echo "$SERVICE is running."
fi
```

Bash Problem 5

Automate the generation of SSL certificates with OpenSSL.

```
#!/bin/bash

DOMAIN="example.com"
CERT_DIR="./certs"
mkdir -p $CERT_DIR

openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
    -keyout "$CERT_DIR/$DOMAIN.key" \
    -out "$CERT_DIR/$DOMAIN.crt" \
    -subj "/C=US/ST=State/L=City/O=Organization/CN=$DOMAIN"

echo "SSL certificate and key generated in $CERT_DIR."
```

Bash Problem 6

Monitor memory usage and log if it exceeds a threshold.

```
#!/bin/bash

THRESHOLD=80
LOGFILE="memory_usage.log"

memory_usage=$(free | awk '/Mem:/ {print $3/$2 * 100.0}')
if (( $(echo "$memory_usage > $THRESHOLD" | bc -l) )); then
    echo "Memory usage is above $THRESHOLD%: $memory_usage%" | tee -a $LOGFILE
else
    echo "Memory usage is normal: $memory_usage%"
fi
```

Bash Problem 7

List all open network ports and their processes.

```
#!/bin/bash

echo "Listing all open network ports and associated processes..."
sudo netstat -tuln | awk 'NR>2 {print $0}'
```

Bash Problem 8

Backup all files in a directory.

```
#!/bin/bash

SOURCE_DIR="/home/user/data"
BACKUP_DIR="/home/user/backup"
```

```
DATE=$(date +%F)

mkdir -p $BACKUP_DIR
tar -czf "$BACKUP_DIR/backup-$DATE.tar.gz" -C $SOURCE_DIR .

echo "Backup completed: $BACKUP_DIR/backup-$DATE.tar.gz"
```

Bash Problem 9

Check for failed SSH login attempts.

```
#!/bin/bash

LOGFILE="/var/log/auth.log"
grep "Failed password" $LOGFILE | awk '{print $1, $2, $3, $11}' | sort | uniq -c
```

Bash Problem 10

Create a script to ping multiple servers and log results.

```
#!/bin/bash

SERVERS=("google.com" "github.com" "example.com")
LOGFILE="ping_results.log"

echo "Ping Test Results:" > $LOGFILE
for server in "${SERVERS[@]}"; do
    if ping -c 1 $server &> /dev/null; then
        echo "$server is reachable." | tee -a $LOGFILE
    else
        echo "$server is unreachable." | tee -a $LOGFILE
    fi
done
```

Bash Problem 11

Automate checking if a specific process is running.

```
#!/bin/bash

PROCESS="nginx"
if pgrep -x "$PROCESS" > /dev/null; then
    echo "$PROCESS is running."
else
    echo "$PROCESS is not running."
fi
```

Bash Problem 12

Delete temporary files older than a specific number of days.

```
#!/bin/bash

DIRECTORY="/tmp"
DAYS=7

find $DIRECTORY -type f -mtime +$DAYS -exec rm -f {} \;
echo "Deleted files older than $DAYS days in $DIRECTORY."
```

Bash Problem 13

Monitor swap usage and log when it exceeds a threshold

```
#!/bin/bash

THRESHOLD=50
LOGFILE="swap_usage.log"

swap_usage=$(free | awk '/Swap/ {print $3/$2 * 100.0}')
if (( $(echo "$swap_usage > $THRESHOLD" | bc -l) )); then
    echo "Swap usage is above $THRESHOLD%: $swap_usage%" | tee -a $LOGFILE
fi
```

Bash Problem 14

Check if a user exists and create it if not.

```
#!/bin/bash

USERNAME="newuser"

if id "$USERNAME" &>/dev/null; then
    echo "User $USERNAME already exists."
else
    sudo useradd $USERNAME
    echo "User $USERNAME created."
fi
```

Bash Problem 15

List all active network connections.

```
#!/bin/bash

echo "Listing all active network connections..."
sudo netstat -tnp | awk 'NR>2 {print $0}'
```

Bash Problem 16

Check the status of all running services and log the ones that are inactive.

```
#!/bin/bash

LOGFILE="inactive_services.log"

echo "Checking for inactive services..." > $LOGFILE
sudo systemctl list-units --type=service --state=inactive | awk '{print $1}' >> $LOGFILE
echo "Inactive services logged in $LOGFILE."
```

Bash Problem 17

Create a script to add multiple users from a text file.

```
#!/bin/bash

USERLIST="users.txt"

if [[ ! -f $USERLIST ]]; then
    echo "User list file not found!"
    exit 1
fi

while IFS= read -r username; do
    if id "$username" &>/dev/null; then
        echo "User $username already exists."
    else
        sudo useradd "$username"
        echo "User $username added."
    fi
done < "$USERLIST"
```

Bash Problem 18

Backup MySQL databases.

```
#!/bin/bash

BACKUP_DIR="/backups"
DB_USER="root"
DB_PASSWORD="password"
DATE=$(date +%F)

mkdir -p $BACKUP_DIR
databases=$(mysql -u$DB_USER -p$DB_PASSWORD -e 'SHOW DATABASES;' | grep -Ev '(Database|information_schema|performance_schema)')

for db in $databases; do
```

```
    mysqldump -u$DB_USER -p$DB_PASSWORD $db > "$BACKUP_DIR/$db-$DATE.sql"
    echo "Database $db backed up."
done
```

Bash Problem 19

Find and delete empty directories in a given path.

```
#!/bin/bash

TARGET_DIR="/tmp"

find $TARGET_DIR -type d -empty -exec rmdir {} \;
echo "Empty directories in $TARGET_DIR have been removed."
```

Bash Problem 20

Generate a list of installed packages.

```
#!/bin/bash

echo "Listing all installed packages..."
dpkg --get-selections | awk '{print $1}' > installed_packages.txt
echo "Package list saved to installed_packages.txt."
```

Bash Problem 21

Find and list all files modified in the last 7 days.

```
#!/bin/bash

DIRECTORY="/var/log"
find $DIRECTORY -type f -mtime -7 -print
```

Bash Problem 22

Automate system reboot if CPU usage exceeds a threshold.

```
#!/bin/bash

CPU_THRESHOLD=90

cpu_usage=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100 - $1}')
if (( $(echo "$cpu_usage > $CPU_THRESHOLD" | bc -l) )); then
    echo "CPU usage exceeded threshold. Rebooting the system..."
    sudo reboot
fi
```

Bash Problem 23

Create a script to check for large files and log them.

```
#!/bin/bash

TARGET_DIR="/home/user"
LOG_FILE="large_files.log"
SIZE_THRESHOLD=1000000  # 1MB

find $TARGET_DIR -type f -size +${SIZE_THRESHOLD}c -exec ls -lh {} \; | awk '{print $9 ": " $5}' >> $LOG_FILE
echo "Large files logged in $LOG_FILE."
```

Bash Problem 24

Send a system resource report via email using `mail` command.

```
#!/bin/bash

CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100 - $1}')
MEMORY_USAGE=$(free | grep Mem | awk '{print $3/$2 * 100.0}')
DISK_USAGE=$(df / | grep / | awk '{ print $5 }')

REPORT="System Resource Report:\nCPU Usage: $CPU_USAGE%\nMemory Usage: $MEMORY_USAGE%\nDisk Usage: $DISK_USAGE"
```

```
echo -e $REPORT | mail -s "System Resource Report" admin@example.com
```

Bash Problem 25

Backup all files from a directory to a remote server using SCP.

```
#!/bin/bash

LOCAL_DIR="/home/user/data"
REMOTE_SERVER="user@remote:/backup"

scp -r $LOCAL_DIR $REMOTE_SERVER
echo "Backup completed to $REMOTE_SERVER."
```

Bash Problem 26

Create a cron job to run a backup script every day at midnight.

```
#!/bin/bash

BACKUP_SCRIPT="/path/to/backup.sh"
crontab -l | { cat; echo "0 0 * * * $BACKUP_SCRIPT"; } | crontab -
echo "Cron job to run backup script at midnight added."
```

Bash Problem 27

Monitor a log file for new entries and print them in real-time.

```
#!/bin/bash

LOG_FILE="/var/log/syslog"
tail -f $LOG_FILE
```

Bash Problem 28

Create a script to rotate log files based on size.

```
#!/bin/bash

LOG_FILE="/var/log/myapp.log"
MAX_SIZE=5000000  # 5MB

FILE_SIZE=$(stat -c %s "$LOG_FILE")

if [ "$FILE_SIZE" -gt "$MAX_SIZE" ]; then
    mv "$LOG_FILE" "$LOG_FILE.old"
    touch "$LOG_FILE"
    echo "Log file rotated."
fi
```

Bash Problem 29

Create a script to monitor the system load every minute and log it to a file.

```
#!/bin/bash

LOG_FILE="system_load.log"
while true; do
    uptime >> $LOG_FILE
    sleep 60  # Monitor every minute
done
```

Bash Problem 30

Check the status of a specific service and restart it if stopped.

```
#!/bin/bash

SERVICE="nginx"
if ! systemctl is-active --quiet $SERVICE; then
    echo "$SERVICE is not running. Restarting..."
    sudo systemctl restart $SERVICE
```

```
else
    echo "$SERVICE is running."
fi
```

Bash Problem 31

Create a script that checks if a specific port is open on the server.

```
#!/bin/bash

PORT=80
HOST="localhost"

nc -zv $HOST $PORT
if [ $? -eq 0 ]; then
    echo "Port $PORT is open."
else
    echo "Port $PORT is closed."
fi
```

Bash Problem 32

Create a script to set file permissions recursively in a directory.

```
#!/bin/bash

DIRECTORY="/home/user/data"
chmod -R 755 $DIRECTORY
echo "Permissions set to 755 for all files in $DIRECTORY."
```

Bash Problem 33

Create a script that sends a system reboot command if the disk usage exceeds a threshold.

```
#!/bin/bash

THRESHOLD=90
DISK_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ "$DISK_USAGE" -gt "$THRESHOLD" ]; then
    echo "Disk usage exceeded $THRESHOLD%. Rebooting system..."
    sudo reboot
fi
```

Bash Problem 34

Monitor the CPU temperature and log it to a file.

```
#!/bin/bash

LOG_FILE="cpu_temp.log"
while true; do
    TEMP=$(cat /sys/class/thermal/thermal_zone0/temp)
    TEMP_C=$((TEMP / 1000))  # Convert to Celsius
    echo "CPU Temperature: $TEMP_C°C" >> $LOG_FILE
    sleep 60  # Log every minute
done
```

Bash Problem 35

Create a script to check if a directory exists, and if not, create it.

```
#!/bin/bash

DIRECTORY="/home/user/data"
if [ ! -d "$DIRECTORY" ]; then
    mkdir -p "$DIRECTORY"
    echo "Directory $DIRECTORY created."
else
    echo "Directory $DIRECTORY already exists."
fi
```

Bash Problem 36

Create a script to monitor a process and log its status.

```
#!/bin/bash

PROCESS="nginx"
LOG_FILE="process_monitor.log"

if pgrep -x "$PROCESS" > /dev/null; then
    echo "$(date): $PROCESS is running." >> $LOG_FILE
else
    echo "$(date): $PROCESS is not running." >> $LOG_FILE
fi
```

Bash Problem 37

Create a script to count the number of lines in a log file.

```
#!/bin/bash

LOG_FILE="/var/log/syslog"
LINE_COUNT=$(wc -l < "$LOG_FILE")
echo "Total lines in $LOG_FILE: $LINE_COUNT"
```

Bash Problem 38

Create a script to archive and compress log files older than a specified number of days.

```
#!/bin/bash

LOG_DIR="/var/log"
ARCHIVE_DIR="/backup/logs"
DAYS_OLD=7
find $LOG_DIR -type f -name "*.log" -mtime +$DAYS_OLD -exec tar -zcvf "$ARCHIVE_DIR/$(basename {}).tar.gz" {} \;
echo "Archived and compressed log files older than $DAYS_OLD days."
```

Bash Problem 39

Create a script to automatically update all installed packages on the system.

```
#!/bin/bash

echo "Updating all packages..."
sudo apt update && sudo apt upgrade -y
echo "Packages updated."
```

Bash Problem 40

Create a script that checks if the system is under heavy load and sends an alert.

```
#!/bin/bash

LOAD_THRESHOLD=2.0
LOAD=$(uptime | awk '{print $(NF-2)}' | sed 's/,//')

if (( $(echo "$LOAD > $LOAD_THRESHOLD" | bc -l) )); then
    echo "System under heavy load! Load: $LOAD" | mail -s "Load Alert" admin@example.com
else
    echo "System load is normal. Load: $LOAD"
fi
```

Bash Problem 41

Create a script that checks for active network interfaces and logs them.

```
#!/bin/bash

echo "Active network interfaces:" > network_interfaces.log
for iface in $(ls /sys/class/net/); do
    if ip link show $iface | grep -q "state UP"; then
        echo "$iface is active" >> network_interfaces.log
    fi
```

```
done
echo "Network interfaces logged to network_interfaces.log."
```

Bash Problem 42

Create a script to count the number of running processes.

```
#!/bin/bash

PROCESS_COUNT=$(ps aux | wc -l)
echo "Number of running processes: $PROCESS_COUNT"
```

Bash Problem 43

Create a script to check disk space and send an alert if it exceeds a certain threshold.

```
#!/bin/bash

THRESHOLD=80
DISK_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ "$DISK_USAGE" -gt "$THRESHOLD" ]; then
    echo "Disk usage exceeded $THRESHOLD%. Sending alert..."
    echo "Disk usage alert: $DISK_USAGE%" | mail -s "Disk Usage Alert" admin@example.com
else
    echo "Disk usage is normal."
fi
```

Bash Problem 44

Create a script to check and log the status of multiple services.

```
#!/bin/bash

SERVICES=("nginx" "apache2" "mysql")
LOG_FILE="service_status.log"

for SERVICE in "${SERVICES[@]}"; do
    if systemctl is-active --quiet $SERVICE; then
        echo "$(date): $SERVICE is running." >> $LOG_FILE
    else
        echo "$(date): $SERVICE is not running." >> $LOG_FILE
    fi
done
echo "Service status logged."
```

Bash Problem 45

Create a script to create a backup of a directory.

```
#!/bin/bash

SOURCE_DIR="/home/user/data"
BACKUP_DIR="/backup/data"
TIMESTAMP=$(date +'%Y%m%d%H%M%S')
BACKUP_FILE="$BACKUP_DIR/data_backup_$TIMESTAMP.tar.gz"

tar -czf $BACKUP_FILE $SOURCE_DIR
echo "Backup created: $BACKUP_FILE"
```

Bash Problem 46

Create a script to rename multiple files in a directory in bulk.

```
#!/bin/bash

DIRECTORY="/home/user/files"
for FILE in $DIRECTORY/*.txt; do
    mv "$FILE" "${FILE%.txt}_new.txt"
done
echo "Files renamed."
```

Bash Problem 47

Create a script to log the system's uptime.

```
#!/bin/bash

UPTIME=$(uptime -p)
echo "System Uptime: $UPTIME" > uptime.log
echo "Uptime logged."
```

Bash Problem 48

Create a script to search for a string in all files in a directory.

```
#!/bin/bash

SEARCH_DIR="/home/user/docs"
SEARCH_STRING="error"
grep -r "$SEARCH_STRING" "$SEARCH_DIR"
```

Bash Problem 49

Create a script to list all running processes and their memory usage.

```
#!/bin/bash

ps aux --sort=-%mem | awk 'NR<=10{print $0}'
```

Bash Problem 50

Create a script to check if a specific user is logged in.

```
#!/bin/bash

USER_TO_CHECK="john"
if who | grep -q "$USER_TO_CHECK"; then
    echo "$USER_TO_CHECK is logged in."
else
    echo "$USER_TO_CHECK is not logged in."
fi
```

Bash Problem 51

Create a script to check the available free memory on the system.

```
#!/bin/bash

FREE_MEMORY=$(free -h | grep Mem | awk '{print $4}')
echo "Free memory: $FREE_MEMORY"
```

Bash Problem 52

Create a script to kill a process by its name

```
#!/bin/bash

PROCESS_NAME="nginx"
PID=$(pgrep -x "$PROCESS_NAME")
if [ -n "$PID" ]; then
    kill -9 $PID
    echo "$PROCESS_NAME process killed."
else
    echo "$PROCESS_NAME is not running."
fi
```

Bash Problem 53

Create a script to check if a specific process is running.

```
#!/bin/bash

PROCESS_NAME="apache2"
if pgrep -x "$PROCESS_NAME" > /dev/null; then
```

```
    echo "$PROCESS_NAME is running."
else
    echo "$PROCESS_NAME is not running."
fi
```

Bash Problem 54

Create a script to check the status of disk space and alert if space is below a certain threshold.

```
#!/bin/bash

THRESHOLD=10
DISK_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')

if [ "$DISK_USAGE" -gt "$THRESHOLD" ]; then
    echo "Warning: Disk space is running low. Usage: $DISK_USAGE%" | mail -s "Disk Space Alert" admin@example.com
else
    echo "Disk space usage is under control. Usage: $DISK_USAGE%"
fi
```

Bash Problem 55

Create a script to show the most memory-consuming processes.

```
#!/bin/bash

ps aux --sort=-%mem | head -n 10
```

Bash Problem 56

Create a script to check for updates on the system and install them.

```
#!/bin/bash

echo "Checking for updates..."
sudo apt update && sudo apt upgrade -y
echo "Updates installed."
```

Bash Problem 57

Create a script to find the largest file in a directory.

```
#!/bin/bash

DIRECTORY="/home/user"
LARGEST_FILE=$(find $DIRECTORY -type f -exec du -h {} + | sort -rh | head -n 1)
echo "Largest file: $LARGEST_FILE"
```

Bash Problem 58

Create a script to automate cleaning of old log files in a directory.

```
#!/bin/bash

LOG_DIR="/var/log"
find $LOG_DIR -type f -name "*.log" -mtime +30 -exec rm -f {} \;
echo "Old log files cleaned."
```

Bash Problem 59

Create a script to search for a string within all files in a directory.

```
#!/bin/bash

DIRECTORY="/home/user"
STRING_TO_SEARCH="error"
grep -r "$STRING_TO_SEARCH" "$DIRECTORY"
```

Bash Problem 60

Create a script to check if a service is running and restart it if necessary.

```
#!/bin/bash

SERVICE_NAME="nginx"
if systemctl is-active --quiet "$SERVICE_NAME"; then
    echo "$SERVICE_NAME is running."
else
    echo "$SERVICE_NAME is not running. Restarting..."
    sudo systemctl restart "$SERVICE_NAME"
fi
```

Bash Problem 61

Create a script to display the current disk usage.

```
#!/bin/bash

df -h
```

Bash Problem 62

Create a script to monitor disk usage and send an alert if it exceeds a threshold.

```
#!/bin/bash

DISK_USAGE=$(df / | grep / | awk '{ print $5 }' | sed 's/%//g')
THRESHOLD=90

if [ "$DISK_USAGE" -gt "$THRESHOLD" ]; then
    echo "Disk usage is over threshold: $DISK_USAGE%" | mail -s "Disk Usage Alert" admin@example.com
else
    echo "Disk usage is under control: $DISK_USAGE%"
fi
```

Bash Problem 63

Create a script to compress log files older than 30 days.

```
#!/bin/bash

LOG_DIR="/var/log"
find $LOG_DIR -type f -name "*.log" -mtime +30 -exec gzip {} \;
echo "Old log files compressed."
```

Bash Problem 64

Create a script to list all running processes.

```
#!/bin/bash

ps aux
```

Bash Problem 65

Create a script to show all the users currently logged in.

```
#!/bin/bash

who
```

Bash Problem 66

Create a script to display the memory usage of the system.

```
#!/bin/bash

free -h
```

Bash Problem 67

Create a script to search for files by name in a directory.

```
#!/bin/bash

DIRECTORY="/home/user"
FILENAME="test.txt"
find $DIRECTORY -type f -name "$FILENAME"
```

Bash Problem 68

Create a script to check if a directory exists and create it if it doesn't.

```
#!/bin/bash

DIRECTORY="/home/user/new_directory"
if [ ! -d "$DIRECTORY" ]; then
    mkdir -p "$DIRECTORY"
    echo "Directory created."
else
    echo "Directory already exists."
fi
```

Bash Problem 69

Create a script to display disk space usage for each mounted filesystem

```
#!/bin/bash

df -h
```

Bash Problem 70

Create a script to create a tar archive of all `.log` files in a directory

```
#!/bin/bash

DIRECTORY="/var/log"
tar -czf logs_backup.tar.gz $DIRECTORY/*.log
echo "Logs have been backed up."
```

Bash Problem 71

Create a script to list all files and directories with their permissions.

```
#!/bin/bash

ls -l
```

Bash Problem 72

Create a script to rename a file or directory.

```
#!/bin/bash

OLD_NAME="old_file.txt"
NEW_NAME="new_file.txt"
mv $OLD_NAME $NEW_NAME
echo "File renamed to $NEW_NAME."
```

Bash Problem 73

Create a script to count the number of files in a directory.

```
#!/bin/bash

DIRECTORY="/home/user"
FILE_COUNT=$(find $DIRECTORY -type f | wc -l)
echo "Number of files: $FILE_COUNT"
```

Bash Problem 74

Create a script to check if a file exists and display its size.

```
#!/bin/bash

FILE="/home/user/file.txt"
if [ -e "$FILE" ]; then
    FILE_SIZE=$(stat -c %s "$FILE")
    echo "$FILE exists. Size: $FILE_SIZE bytes."
else
    echo "$FILE does not exist."
fi
```

Bash Problem 75

Create a script to show the current system uptime.

```
#!/bin/bash

uptime
```

Bash Problem 76

Create a script to list all active network connections.

```
#!/bin/bash

netstat -tuln
```

Bash Problem 77

Create a script to display the current date and time.

```
#!/bin/bash

date
```

Bash Problem 78

Create a script to check if a service is running and restart it if it is not.

```
#!/bin/bash

SERVICE="nginx"
if ! systemctl is-active --quiet $SERVICE; then
    echo "$SERVICE is not running. Restarting..."
    sudo systemctl restart $SERVICE
else
    echo "$SERVICE is running."
fi
```

Bash Problem 79

Create a script to show the top 10 most memory-consuming processes.

```
#!/bin/bash

ps aux --sort=-%mem | head -n 10
```

Bash Problem 80

Create a script to display the disk usage of each user on the system.

```
#!/bin/bash

du -sh /home/*
```

Bash Problem 81

Create a script to monitor system load and send an email alert if it exceeds a threshold.

```
#!/bin/bash

LOAD_THRESHOLD=2.0
current_load=$(uptime | awk '{print $10}' | sed 's/,//')
```

```
if (( $(echo "$current_load > $LOAD_THRESHOLD" | bc -l) )); then
    echo "System load is high: $current_load" | mail -s "System Load Alert" admin@example.com
fi
```

Bash Problem 82

Create a script to search for a string within files and display matching lines.

```
#!/bin/bash

STRING="error"
DIRECTORY="/home/user/logs"
grep -r "$STRING" $DIRECTORY
```

Bash Problem 83

Create a script to display the last 10 lines of a log file.

```
#!/bin/bash

LOG_FILE="/var/log/syslog"
tail -n 10 $LOG_FILE
```

Bash Problem 84

Create a script to monitor a file's size and alert if it exceeds a threshold.

```
#!/bin/bash

FILE="/home/user/log.txt"
THRESHOLD=5000
FILE_SIZE=$(stat -c %s "$FILE")

if [ $FILE_SIZE -gt $THRESHOLD ]; then
    echo "File size exceeds threshold: $FILE_SIZE bytes" | mail -s "File Size Alert" admin@example.com
fi
```

Bash Problem 85

Create a script to check disk space usage for all mounted filesystems.

```
#!/bin/bash

df -h
```

Bash Problem 86

Create a script to monitor CPU usage and alert if it exceeds a threshold.

```
#!/bin/bash

CPU_THRESHOLD=90
CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100 - $1}')
if (( $(echo "$CPU_USAGE > $CPU_THRESHOLD" | bc -l) )); then
    echo "CPU usage is high: $CPU_USAGE%" | mail -s "CPU Usage Alert" admin@example.com
fi
```

Bash Problem 87

Create a script to find and delete all empty files in a directory.

```
#!/bin/bash

find /home/user/directory -type f -empty -exec rm {} \;
```

Bash Problem 88

Create a script to display the current memory usage in a human-readable format.

```
#!/bin/bash
```

```
free -h
```

Bash Problem 89

Create a script to check if a file exists and display an appropriate message.

```
#!/bin/bash

FILE="/home/user/file.txt"
if [ -e "$FILE" ]; then
    echo "File exists."
else
    echo "File does not exist."
fi
```

Bash Problem 90

Create a script to display system load and alert if it exceeds a threshold.

```
#!/bin/bash

LOAD_THRESHOLD=2.0
current_load=$(uptime | awk '{print $10}' | sed 's/,//')

if (( $(echo "$current_load > $LOAD_THRESHOLD" | bc -l) )); then
    echo "High system load: $current_load" | mail -s "System Load Alert" admin@example.com
fi
```

Bash Problem 91

Create a script to list all running processes with detailed information.

```
#!/bin/bash

ps aux
```

Bash Problem 92

Create a script to display the system's current date and time in a specific format.

```
#!/bin/bash

date +"%Y-%m-%d %H:%M:%S"
```

Bash Problem 93

Create a script to backup files from one directory to another.

```
#!/bin/bash

SOURCE_DIR="/home/user/data"
BACKUP_DIR="/home/user/backup"

cp -r $SOURCE_DIR/* $BACKUP_DIR/
echo "Backup completed."
```

Bash Problem 94

Create a script to change file permissions recursively in a directory.

```
#!/bin/bash

DIRECTORY="/home/user/documents"
chmod -R 755 $DIRECTORY
echo "Permissions changed recursively."
```

Bash Problem 95

Create a script to download a file from a URL.

```
#!/bin/bash

URL="http://example.com/file.txt"
```

```
OUTPUT="/home/user/file.txt"

curl -o $OUTPUT $URL
echo "Download completed."
```

Bash Problem 96

Create a script to list all users who are currently logged in.

```
#!/bin/bash

who
```

Bash Problem 97

Create a script to list files older than a certain number of days and delete them

```
#!/bin/bash

DIRECTORY="/home/user/old_files"
DAYS=30

find $DIRECTORY -type f -mtime +$DAYS -exec rm {} \;
echo "Old files deleted."
```

Bash Problem 98

Create a script to check the system's uptime.

```
#!/bin/bash

uptime -p
```

Bash Problem 99

Create a script to create a new user and set a password.

```
#!/bin/bash

USER="newuser"
PASSWORD="password123"

useradd $USER
echo "$USER:$PASSWORD" | chpasswd
echo "User $USER created with password."
```

Bash Problem 100

Create a script to monitor system load and send an email alert if it exceeds a threshold.

```
#!/bin/bash

LOAD_THRESHOLD=2.5
LOAD=$(uptime | awk '{print $10}' | sed 's/,//')

if (( $(echo "$LOAD > $LOAD_THRESHOLD" | bc -l) )); then
    echo "High system load: $LOAD" | mail -s "System Load Alert" admin@example.com
fi
```