**1. Hello World Script**

bash

CopyEdit

#!/bin/bash

echo "Hello, World!"

* **#!/bin/bash**: This “shebang” tells the system to execute the script using the Bash shell.
* **echo "Hello, World!"**: Prints the text "Hello, World!" to the terminal.

**2. Print Numbers from 1 to 5**

bash

CopyEdit

#!/bin/bash

for i in {1..5}

do

echo $i

done

* **for i in {1..5}**: Creates a loop where variable i takes values from 1 to 5.
* **echo $i**: Prints the current number during each loop iteration.

**3. Check if a File Exists**

bash

CopyEdit

#!/bin/bash

if [ -f "$1" ]; then

echo "File exists."

else

echo "File does not exist."

fi

* **[ -f "$1" ]**: Checks whether the first argument is a regular file.
* **if/then/else/fi**: Standard conditional structure to output a message based on whether the file exists.

**4. Count Files in a Directory**

bash

CopyEdit

#!/bin/bash

echo "Number of files in $1: $(ls -1 "$1" | wc -l)"

* **ls -1 "$1"**: Lists the files (one per line) in the specified directory.
* **wc -l**: Counts the number of lines (files) output by ls.

**5. Backup a Directory**

bash

CopyEdit

#!/bin/bash

tar -czvf backup\_$(date +%F).tar.gz "$1"

echo "Backup created: backup\_$(date +%F).tar.gz"

* **tar -czvf**: Archives (tar) the target directory, compresses it with gzip (z), shows progress (v), and names the output file (f).
* **$(date +%F)**: Inserts the current date in YYYY-MM-DD format, ensuring unique backup filenames.

**6. Monitor Disk Usage**

bash

CopyEdit

#!/bin/bash

df -h

* **df -h**: Displays disk space usage in a human-readable format (e.g., MB/GB).

**7. Send an Email Notification (using mail)**

bash

CopyEdit

#!/bin/bash

echo "This is a test email" | mail -s "Test Email" user@example.com

* **echo "This is a test email"**: Provides the body of the email.
* **mail -s "Test Email" user@example.com**: Sends the email with the subject "Test Email" to the specified recipient.

**8. Download a File Using wget**

bash

CopyEdit

#!/bin/bash

wget "$1"

* **wget "$1"**: Downloads a file from the URL provided as the first argument.

**9. Archive Logs Using tar**

bash

CopyEdit

#!/bin/bash

tar -czvf logs\_backup\_$(date +%F).tar.gz /var/log

* **tar -czvf**: Creates a compressed archive of the /var/log directory, naming the file with the current date.

**10. Start a Service if Not Running**

bash

CopyEdit

#!/bin/bash

if ! systemctl is-active --quiet "$1"; then

systemctl start "$1"

echo "$1 started."

else

echo "$1 is already running."

fi

* **systemctl is-active --quiet "$1"**: Checks if the given service is active without printing output.
* **systemctl start "$1"**: Starts the service if it isn’t running.
* **Conditional block**: Ensures the service is only started when needed.

**11. Stop a Service**

bash

CopyEdit

#!/bin/bash

systemctl stop "$1"

echo "$1 stopped."

* **systemctl stop "$1"**: Stops the specified service.
* **echo "$1 stopped."**: Confirms the action.

**12. Check System Load**

bash

CopyEdit

#!/bin/bash

uptime

* **uptime**: Displays how long the system has been running along with the current load averages.

**13. Monitor CPU Usage**

bash

CopyEdit

#!/bin/bash

top -b -n1 | head -n 10

* **top -b -n1**: Runs top in batch mode for one iteration to capture system information.
* **head -n 10**: Displays the first 10 lines of output (usually showing summary info and the top processes).

**14. Monitor Memory Usage**

bash

CopyEdit

#!/bin/bash

free -m

* **free -m**: Displays memory usage statistics in megabytes.

**15. Monitor Network Connectivity (Ping)**

bash

CopyEdit

#!/bin/bash

ping -c 4 "$1"

* **ping -c 4 "$1"**: Sends 4 ICMP echo requests to the provided host, testing network connectivity.

**16. Grep Error Logs and Send Email Alert**

bash

CopyEdit

#!/bin/bash

if grep -i "error" /var/log/syslog; then

echo "Errors found in syslog." | mail -s "Syslog Errors" admin@example.com

fi

* **grep -i "error" /var/log/syslog**: Searches the syslog for the word “error” (case-insensitive).
* **mail**: Sends an email alert if errors are detected.

**17. Add a Cron Job (Scheduling Example)**

bash

CopyEdit

#!/bin/bash

echo "0 2 \* \* \* /path/to/script.sh" | crontab -

* **echo "0 2 \* \* \* /path/to/script.sh"**: Defines a cron job to run a script daily at 2:00 AM.
* **crontab -**: Installs the cron job for the current user.

**18. Delete Old Files Based on Age**

bash

CopyEdit

#!/bin/bash

find "$1" -type f -mtime +7 -exec rm {} \;

* **find "$1" -type f -mtime +7**: Searches for files in the specified directory that were modified more than 7 days ago.
* **-exec rm {} ;**: Deletes each file found.

**19. Search and Replace Text in Files**

bash

CopyEdit

#!/bin/bash

sed -i 's/old/new/g' "$1"

* **sed -i 's/old/new/g' "$1"**: Uses stream editor sed to replace every occurrence of “old” with “new” directly in the file.

**20. Check if a Process is Running**

bash

CopyEdit

#!/bin/bash

if pgrep -x "$1" > /dev/null; then

echo "$1 is running."

else

echo "$1 is not running."

fi

* **pgrep -x "$1"**: Searches for processes with an exact match of the provided name.
* **> /dev/null**: Discards output, using only the exit status to determine if the process is running.

**21. Kill a Process by Name**

bash

CopyEdit

#!/bin/bash

pkill -f "$1"

echo "Killed process: $1"

* **pkill -f "$1"**: Terminates all processes whose command lines match the given pattern.
* **echo**: Confirms the action.

**22. Rotate Log Files**

bash

CopyEdit

#!/bin/bash

mv "$1" "$1.$(date +%F)"

touch "$1"

echo "Log rotated."

* **mv "$1" "$1.$(date +%F)"**: Renames the log file by appending the current date.
* **touch "$1"**: Creates a new, empty file with the original log file’s name.

**23. Create a New User Account**

bash

CopyEdit

#!/bin/bash

sudo useradd -m "$1"

echo "User $1 created."

* **sudo useradd -m "$1"**: Creates a new user with a home directory (-m).
* **echo**: Outputs a confirmation message.

**24. Delete User Accounts Older Than X Days**

bash

CopyEdit

#!/bin/bash

lastlog -b "$1" | awk '$NF < "'$2'" {print $1}' | xargs -I {} sudo userdel -r {}

* **lastlog -b "$1"**: Lists users whose last login was before a certain number of days.
* **awk**: Filters the output to extract usernames based on the specified condition.
* **xargs -I {} sudo userdel -r {}**: Deletes each user (with their home directory) that meets the criteria.

**25. Check if a Port is Open**

bash

CopyEdit

#!/bin/bash

if netstat -tuln | grep ":$1"; then

echo "Port $1 is in use."

else

echo "Port $1 is free."

fi

* **netstat -tuln**: Lists all TCP/UDP ports in a numeric format.
* **grep ":$1"**: Searches for the specified port in the list.

**26. Backup a MySQL Database**

bash

CopyEdit

#!/bin/bash

mysqldump -u root -p"$2" "$1" > "${1}\_backup\_$(date +%F).sql"

echo "Backup of $1 completed."

* **mysqldump**: Exports a MySQL database to a SQL file.
* **-u root -p"$2"**: Uses the root user with the password provided as the second argument.
* **> "${1}*backup*$(date +%F).sql"**: Redirects output to a file named with the database name and current date.

**27. Restore a MySQL Database Backup**

bash

CopyEdit

#!/bin/bash

mysql -u root -p"$2" "$1" < "$3"

echo "Database $1 restored from $3."

* **mysql -u root -p"$2" "$1" < "$3"**: Restores the database by piping in the SQL file.

**28. Monitor a Website’s Availability**

bash

CopyEdit

#!/bin/bash

if curl -s --head "$1" | head -n 1 | grep "200 OK" > /dev/null; then

echo "Website is up."

else

echo "Website is down."

fi

* **curl -s --head "$1"**: Fetches the HTTP headers silently.
* **head -n 1**: Extracts the first header line (which typically contains the status code).
* **grep "200 OK"**: Checks if the HTTP response indicates success.

**29. Parse a CSV File and Display Data**

bash

CopyEdit

#!/bin/bash

awk -F, '{print $1, $2}' "$1"

* **awk -F,**: Sets the field separator to a comma for CSV parsing.
* **'{print $1, $2}'**: Prints the first two columns of each line.

**30. Count Lines in a File**

bash

CopyEdit

#!/bin/bash

wc -l "$1"

* **wc -l**: Counts and prints the number of lines in the file.

**31. Create a Menu-Driven Shell Script**

bash

CopyEdit

#!/bin/bash

while true; do

echo "1. Option One"

echo "2. Option Two"

echo "3. Exit"

read -p "Choose an option: " choice

case $choice in

1) echo "Option One selected";;

2) echo "Option Two selected";;

3) exit;;

\*) echo "Invalid option";;

esac

done

* **while true; do ... done**: Creates an infinite loop to repeatedly show the menu.
* **read -p**: Prompts the user to enter their choice.
* **case/esac**: Branches execution based on user input.

**32. Print System Information**

bash

CopyEdit

#!/bin/bash

uname -a

* **uname -a**: Displays detailed information about the system kernel and architecture.

**33. Display Calendar for a Given Month/Year**

bash

CopyEdit

#!/bin/bash

cal "$@"

* **cal "$@"**: Uses the cal command to display a calendar, passing all script arguments to it.

**34. Simple Calculator**

bash

CopyEdit

#!/bin/bash

read -p "Enter first number: " a

read -p "Enter operator (+, -, \*, /): " op

read -p "Enter second number: " b

result=$(echo "$a $op $b" | bc -l)

echo "Result: $result"

* **read -p**: Prompts the user for input values and the operator.
* **echo "$a $op $b" | bc -l**: Uses the bc command-line calculator to perform the arithmetic with floating point support.
* **$( ... )**: Captures the output of the command substitution into a variable.

**35. Monitor File Changes with inotifywait**

bash

CopyEdit

#!/bin/bash

inotifywait -m "$1"

* **inotifywait -m "$1"**: Uses the inotifywait command to monitor the specified file or directory for changes in a continuous mode (-m).

**36. Convert CSV to JSON (Basic Conversion)**

bash

CopyEdit

#!/bin/bash

awk -F, 'BEGIN {print "["} {printf "{\"col1\":\"%s\", \"col2\":\"%s\"}", $1, $2; if (NR>1) print "},"} END {print "]}"}' "$1"

* **awk -F,**: Sets the CSV delimiter.
* **BEGIN/END Blocks**: Define actions at the start and end of processing (printing JSON array brackets).
* **printf**: Formats each line as a JSON object.

**37. Extract Specific Columns from a CSV File**

bash

CopyEdit

#!/bin/bash

awk -F, '{print $1, $3}' "$1"

* **awk -F,**: Splits each line by commas.
* **'{print $1, $3}'**: Prints the first and third columns.

**38. Generate a Random Password**

bash

CopyEdit

#!/bin/bash

tr -dc 'A-Za-z0-9\_@#%&' </dev/urandom | head -c 12; echo

* **tr -dc 'A-Za-z0-9\_@#%&' </dev/urandom**: Reads from /dev/urandom and deletes all characters except those specified, producing random data.
* **head -c 12**: Limits the output to 12 characters.
* **echo**: Adds a newline at the end.

**39. Parse Command-Line Arguments**

bash

CopyEdit

#!/bin/bash

while [[ "$#" -gt 0 ]]; do

case $1 in

-f|--file) file="$2"; shift ;;

-n|--name) name="$2"; shift ;;

\*) echo "Unknown parameter: $1"; exit 1 ;;

esac

shift

done

echo "File: $file, Name: $name"

* **while [[ "$#" -gt 0 ]]**: Loops through all provided command-line arguments.
* **case $1 in ... esac**: Handles different argument flags (-f, --file, etc.).
* **shift**: Moves to the next argument.

**40. Check for Root Privileges**

bash

CopyEdit

#!/bin/bash

if [ "$EUID" -ne 0 ]; then

echo "Please run as root"

exit 1

fi

echo "Running as root"

* **[ "$EUID" -ne 0 ]**: Checks if the effective user ID is not 0 (root).
* **exit 1**: Exits the script if not running as root.

**41. Update System Packages (Ubuntu)**

bash

CopyEdit

#!/bin/bash

sudo apt update && sudo apt upgrade -y

* **sudo apt update**: Updates the package list.
* **&&**: Executes the next command only if the previous one succeeds.
* **sudo apt upgrade -y**: Upgrades all packages, automatically answering “yes” to prompts.

**42. Check Available Updates (Ubuntu)**

bash

CopyEdit

#!/bin/bash

sudo apt update && apt list --upgradable

* **apt list --upgradable**: Lists packages that have available updates after refreshing the package list.

**43. Monitor Log File Changes in Real Time**

bash

CopyEdit

#!/bin/bash

tail -f "$1"

* **tail -f**: Continuously displays the end of a file (useful for watching log updates).

**44. Get Currently Logged-In Users**

bash

CopyEdit

#!/bin/bash

who

* **who**: Displays information about currently logged-in users.

**45. Find Duplicate Files in a Directory**

bash

CopyEdit

#!/bin/bash

find "$1" -type f -exec md5sum {} + | sort | uniq -w32 -dD

* **find "$1" -type f**: Searches for files in the directory.
* **-exec md5sum {} +**: Computes MD5 checksums for each file.
* **sort | uniq -w32 -dD**: Sorts and identifies duplicates by comparing the first 32 characters of the checksum.

**46. Search Files by Extension**

bash

CopyEdit

#!/bin/bash

find "$1" -name "\*.$2"

* **find "$1" -name "\*.$2"**: Searches for files ending with a specific extension provided as the second argument.

**47. List Top 10 Largest Files**

bash

CopyEdit

#!/bin/bash

find "$1" -type f -exec ls -lh {} + | sort -k5 -rh | head -n 10

* **find "$1" -type f -exec ls -lh {} +**: Lists files with detailed information (including file sizes).
* **sort -k5 -rh**: Sorts the files by the size column (5th field) in reverse (largest first).
* **head -n 10**: Displays the top 10 entries.

**48. Monitor Disk Space and Alert if Low**

bash

CopyEdit

#!/bin/bash

usage=$(df / | tail -1 | awk '{print $5}' | sed 's/%//')

if [ "$usage" -gt 90 ]; then

echo "Disk usage is above 90%!"

fi

* **df /**: Shows disk usage for the root filesystem.
* **tail -1 | awk '{print $5}'**: Extracts the percentage usage from the last line.
* **sed 's/%//'**: Removes the percent sign to compare numerically.
* **if [ "$usage" -gt 90 ]**: Checks if usage exceeds 90%.

**49. AWS CLI Automation (List S3 Buckets)**

bash

CopyEdit

#!/bin/bash

aws s3 ls

* **aws s3 ls**: Uses the AWS CLI to list all S3 buckets associated with the configured account.

**50. Kubernetes Automation (List Pods)**

bash

CopyEdit

#!/bin/bash

kubectl get pods

* **kubectl get pods**: Lists all pods in the current Kubernetes namespace.

**51. Load Environment Variables from a File**

bash

CopyEdit

#!/bin/bash

set -a

source "$1"

set +a

echo "Environment variables loaded."

* **set -a**: Automatically exports all defined variables.
* **source "$1"**: Reads and executes commands from the specified file (typically setting environment variables).
* **set +a**: Disables automatic exporting.

**52. Export System Metrics to a CSV File**

bash

CopyEdit

#!/bin/bash

echo "Time,Load,Memory" > metrics.csv

echo "$(date),$(uptime | awk -F'load average:' '{print $2}'),$(free -m | awk '/Mem/ {print $3}')" >> metrics.csv

* **echo "Time,Load,Memory" > metrics.csv**: Writes a header row to a CSV file.
* **$(date)**, **$(uptime …)**, **$(free -m …)**: Captures current time, system load, and used memory; appends them as a new row.

**53. Automate SVN Checkout/Update**

bash

CopyEdit

#!/bin/bash

svn checkout "$1" || svn update "$1"

* **svn checkout "$1"**: Attempts to check out a repository.
* **|| svn update "$1"**: If checkout fails (perhaps because the repository already exists), it updates the existing copy.

**54. Automate Git Pull for Multiple Repositories**

bash

CopyEdit

#!/bin/bash

for repo in repo1 repo2 repo3; do

cd "/path/to/$repo" && git pull

cd -

done

* **for repo in repo1 repo2 repo3**: Iterates over a list of repository names.
* **cd "/path/to/$repo" && git pull**: Changes to each repository’s directory and pulls the latest changes.
* **cd -**: Returns to the previous directory.

**55. Send a Slack Notification Using curl**

bash

CopyEdit

#!/bin/bash

curl -X POST -H 'Content-type: application/json' \

--data '{"text":"Alert from script"}' \

https://hooks.slack.com/services/your/webhook/url

* **curl -X POST**: Sends an HTTP POST request.
* **-H 'Content-type: application/json'**: Sets the header to indicate JSON data.
* **--data '{"text":"Alert from script"}'**: Provides the JSON payload.
* **Webhook URL**: The endpoint to which Slack listens for incoming messages.

**56. Docker Container Management (List Containers)**

bash

CopyEdit

#!/bin/bash

docker ps -a

* **docker ps -a**: Lists all Docker containers, including both running and stopped ones.

**57. Backup Docker Volumes**

bash

CopyEdit

#!/bin/bash

docker run --rm -v $1:/volume -v $(pwd):/backup alpine tar czf /backup/volume\_backup.tar.gz -C /volume .

* **docker run --rm**: Runs a temporary Docker container and removes it after execution.
* **-v $1:/volume**: Mounts the Docker volume to be backed up.
* **-v $(pwd):/backup**: Mounts the current directory for storing the backup.
* **alpine tar czf …**: Uses the lightweight Alpine image to run tar, creating a gzipped archive.

**58. Monitor Docker Container Health**

bash

CopyEdit

#!/bin/bash

docker ps --filter "health=unhealthy"

* **docker ps --filter "health=unhealthy"**: Filters and lists Docker containers that are reported as unhealthy.

**59. Restart Unhealthy Docker Containers**

bash

CopyEdit

#!/bin/bash

unhealthy=$(docker ps --filter "health=unhealthy" -q)

[ -n "$unhealthy" ] && docker restart $unhealthy

* **$(docker ps --filter "health=unhealthy" -q)**: Captures the IDs of unhealthy containers.
* **[ -n "$unhealthy" ]**: Checks if any unhealthy containers exist.
* **docker restart $unhealthy**: Restarts the listed containers.

**60. Clean Up Stopped Docker Containers**

bash

CopyEdit

#!/bin/bash

docker container prune -f

* **docker container prune -f**: Removes all stopped containers forcefully (without confirmation).

**61. Remove Dangling Docker Images**

bash

CopyEdit

#!/bin/bash

docker image prune -f

* **docker image prune -f**: Removes dangling (unused) Docker images.

**62. Archive Service Logs**

bash

CopyEdit

#!/bin/bash

journalctl -u "$1" > "${1}\_logs\_$(date +%F).log"

* **journalctl -u "$1"**: Fetches logs for a specific systemd service.
* **> "${1}*logs*$(date +%F).log"**: Redirects output to a log file with a timestamp.

**63. Time-Based Job Scheduling (Using sleep)**

bash

CopyEdit

#!/bin/bash

while true; do

echo "Job executed at $(date)"

sleep 3600

done

* **while true; do ... done**: Infinite loop to run a job repeatedly.
* **sleep 3600**: Pauses execution for 3600 seconds (1 hour) between iterations.

**64. Read a Configuration File and Apply Settings**

bash

CopyEdit

#!/bin/bash

source "$1"

echo "Configuration applied."

* **source "$1"**: Reads and executes the contents of the configuration file, importing its variables.

**65. Generate an SSL Certificate with OpenSSL**

bash

CopyEdit

#!/bin/bash

openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout mycert.key -out mycert.crt

* **openssl req -x509**: Generates a self-signed X.509 certificate.
* **-nodes**: Disables passphrase encryption on the key.
* **-days 365**: Sets the certificate validity to 365 days.
* **-newkey rsa:2048**: Creates a new RSA key with a 2048-bit length.
* **-keyout / -out**: Specifies output files for the key and certificate.

**66. Check SSL Certificate Expiration Date**

bash

CopyEdit

#!/bin/bash

echo | openssl s\_client -servername "$1" -connect "$1:443" 2>/dev/null | openssl x509 -noout -dates

* **openssl s\_client**: Connects to the specified server and port (443 for HTTPS).
* **2>/dev/null**: Hides error messages.
* **openssl x509 -noout -dates**: Extracts and displays the certificate’s start and expiry dates.

**67. Parse Apache Logs and Generate a Summary**

bash

CopyEdit

#!/bin/bash

awk '{print $1}' /var/log/apache2/access.log | sort | uniq -c | sort -nr | head

* **awk '{print $1}'**: Extracts the first column (typically the client IP) from Apache’s access log.
* **sort | uniq -c**: Counts occurrences of each IP.
* **sort -nr**: Sorts the counts in descending order.
* **head**: Displays the top results.

**68. Parse Nginx Logs and Generate a Summary**

bash

CopyEdit

#!/bin/bash

awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr | head

* **Similar to Apache log parsing**, this script processes Nginx logs in the same manner.

**69. Monitor System Uptime**

bash

CopyEdit

#!/bin/bash

uptime -p

* **uptime -p**: Displays the system uptime in a human-friendly (“pretty”) format.

**70. Set the System Timezone**

bash

CopyEdit

#!/bin/bash

timedatectl set-timezone "$1"

echo "Timezone set to $1."

* **timedatectl set-timezone "$1"**: Sets the system’s timezone to the specified value.

**71. Change File Permissions Recursively**

bash

CopyEdit

#!/bin/bash

chmod -R "$1" "$2"

echo "Permissions changed."

* **chmod -R**: Recursively changes file permissions for all files and subdirectories within the specified directory.

**72. Find Files with Specific Permissions**

bash

CopyEdit

#!/bin/bash

find "$1" -perm "$2"

* **find "$1" -perm "$2"**: Searches the given directory for files that have the exact permission set specified by the second argument.

**73. Set Up a Python Virtual Environment**

bash

CopyEdit

#!/bin/bash

python3 -m venv env && source env/bin/activate

* **python3 -m venv env**: Creates a new Python virtual environment named env.
* **source env/bin/activate**: Activates the virtual environment so that subsequent Python commands use it.

**74. Install Python Dependencies from requirements.txt**

bash

CopyEdit

#!/bin/bash

source env/bin/activate && pip install -r requirements.txt

* **pip install -r requirements.txt**: Installs all Python packages listed in requirements.txt within the activated environment.

**75. Check if a Python Virtual Environment Exists**

bash

CopyEdit

#!/bin/bash

[ -d "env" ] && echo "Virtualenv exists" || echo "Virtualenv does not exist"

* **[ -d "env" ]**: Tests whether the env directory exists.
* **&& / ||**: Conditional operators that echo the appropriate message based on the test result.

**76. Automate Log Rotation for Custom Logs**

bash

CopyEdit

#!/bin/bash

mv "$1" "$1.$(date +%F)" && touch "$1"

echo "Log rotated."

* **mv "$1" "$1.$(date +%F)"**: Renames the log file by appending the current date.
* **touch "$1"**: Creates a new, empty log file with the original name.

**77. Monitor a Service and Restart if It’s Down**

bash

CopyEdit

#!/bin/bash

service="$1"

if ! systemctl is-active --quiet "$service"; then

systemctl restart "$service"

echo "$service restarted."

fi

* **systemctl is-active --quiet "$service"**: Checks if the service is active.
* **systemctl restart "$service"**: Restarts the service if it is not active.

**78. Daily Backup of Important Configuration Files**

bash

CopyEdit

#!/bin/bash

tar -czvf config\_backup\_$(date +%F).tar.gz /etc

echo "Configuration backup complete."

* **tar -czvf**: Archives and compresses the /etc directory (which typically holds configuration files).

**79. Check the Status of a systemd Service**

bash

CopyEdit

#!/bin/bash

systemctl status "$1"

* **systemctl status "$1"**: Displays detailed status information about the specified service.

**80. Check if a File is Locked by Another Process**

bash

CopyEdit

#!/bin/bash

lsof "$1" || echo "File not in use."

* **lsof "$1"**: Lists open files associated with the specified file, showing which processes may be using it.
* **|| echo "File not in use."**: If no process is found using the file, it outputs a message indicating the file is free.

**81. Automate SSH Key Distribution**

bash

CopyEdit

#!/bin/bash

ssh-copy-id -i "$HOME/.ssh/id\_rsa.pub" "$1"

* **ssh-copy-id**: Copies the public SSH key to a remote host, enabling key-based authentication.

**82. Update /etc/hosts with New Entries**

bash

CopyEdit

#!/bin/bash

echo "$1 $2" >> /etc/hosts

echo "Entry added to /etc/hosts."

* **echo "$1 $2" >> /etc/hosts**: Appends a new hostname-to-IP mapping (or vice versa) to the /etc/hosts file.

**83. Check and Remove Broken Symbolic Links**

bash

CopyEdit

#!/bin/bash

find "$1" -xtype l -delete

echo "Broken links removed."

* **find "$1" -xtype l**: Searches for symbolic links that are broken.
* **-delete**: Removes each broken link found.

**84. Get Hardware Information (CPU, Memory, Disk)**

bash

CopyEdit

#!/bin/bash

lscpu; free -h; lsblk

* **lscpu**: Displays detailed CPU architecture information.
* **free -h**: Shows memory usage in a human-readable format.
* **lsblk**: Lists block devices (disks and partitions).

**85. Monitor System Temperature (with lm-sensors)**

bash

CopyEdit

#!/bin/bash

sensors

* **sensors**: Reads and displays sensor information such as temperatures, voltages, and fan speeds.

**86. Display a Welcome Message at Login**

bash

CopyEdit

#!/bin/bash

echo "Welcome, $(whoami)! Have a great day!"

* **whoami**: Retrieves the current user's name.
* **echo**: Displays a personalized welcome message.

**87. Schedule a System Reboot in 60 Minutes**

bash

CopyEdit

#!/bin/bash

shutdown -r +60

echo "System will reboot in 60 minutes."

* **shutdown -r +60**: Schedules a system reboot after 60 minutes.

**88. Run Multiple Commands in Parallel**

bash

CopyEdit

#!/bin/bash

command1 & command2 & wait

echo "Both commands completed."

* **command1 & command2 &**: Runs command1 and command2 in the background concurrently.
* **wait**: Pauses the script until all background processes complete.

**89. Measure Script Execution Time**

bash

CopyEdit

#!/bin/bash

start=$(date +%s)

# ... your commands here ...

end=$(date +%s)

echo "Execution time: $((end - start)) seconds"

* **$(date +%s)**: Captures the current time in seconds since the Unix epoch.
* **$((end - start))**: Calculates the total elapsed time.

**90. Generate a System Inventory Report**

bash

CopyEdit

#!/bin/bash

{

echo "CPU:"; lscpu

echo "Memory:"; free -h

echo "Disk:"; lsblk

} > inventory\_$(date +%F).txt

* **{ ... } > file**: Groups multiple commands and redirects their combined output to a file.
* **lscpu, free -h, lsblk**: Collect hardware information, which is written to a timestamped file.

**91. Batch Convert Video Files Using ffmpeg**

bash

CopyEdit

#!/bin/bash

for file in \*.mp4; do

ffmpeg -i "$file" "${file%.mp4}.avi"

done

* \**for file in .mp4*: Loops over all .mp4 files in the directory.
* **ffmpeg -i "$file" "${file%.mp4}.avi"**: Converts each video file from MP4 to AVI format, using parameter expansion to change the file extension.

**92. Monitor Process Memory Usage and Kill if Excessive**

bash

CopyEdit

#!/bin/bash

pid=$(pgrep "$1")

mem=$(ps -o %mem= -p $pid)

if (( $(echo "$mem > 50" | bc -l) )); then

kill $pid

echo "Process $1 killed due to high memory usage."

fi

* **pgrep "$1"**: Retrieves the process ID (PID) of the process named in the argument.
* **ps -o %mem= -p $pid**: Extracts the percentage of memory used by the process.
* **bc -l**: Performs a floating point comparison; if memory usage is greater than 50%, the process is killed.

**93. Clean Up Temporary Files Older Than 7 Days**

bash

CopyEdit

#!/bin/bash

find /tmp -type f -atime +7 -delete

echo "Temporary files cleaned."

* **find /tmp -type f -atime +7**: Searches the /tmp directory for files accessed more than 7 days ago.
* **-delete**: Deletes each file found.

**94. Check if a Service is Enabled at Boot**

bash

CopyEdit

#!/bin/bash

systemctl is-enabled "$1"

* **systemctl is-enabled "$1"**: Checks whether the specified service is configured to start automatically on boot.

**95. Monitor Network Bandwidth Usage (using ifstat)**

bash

CopyEdit

#!/bin/bash

ifstat -i "$1" 1 5

* **ifstat -i "$1" 1 5**: Monitors network bandwidth on the specified interface, taking measurements at 1-second intervals over 5 iterations.

**96. Retrieve Public IP Address Using curl**

bash

CopyEdit

#!/bin/bash

curl -s https://api.ipify.org

* **curl -s https://api.ipify.org**: Silently fetches the public IP address from the ipify API.

**97. Automate Package Installations for a New Server (Ubuntu)**

bash

CopyEdit

#!/bin/bash

sudo apt update && sudo apt install -y git curl vim

* **sudo apt update**: Refreshes the package list.
* **sudo apt install -y git curl vim**: Installs Git, curl, and vim without prompting for confirmation.

**98. Monitor File Integrity Using md5sum**

bash

CopyEdit

#!/bin/bash

md5sum "$1" > "$1.md5"

echo "MD5 hash generated for $1."

* **md5sum "$1"**: Calculates the MD5 checksum of the specified file.
* **> "$1.md5"**: Redirects the checksum output to a new file named after the original file with a .md5 extension.

**99. Parse JSON Response Using jq**

bash

CopyEdit

#!/bin/bash

curl -s "$1" | jq .

* **curl -s "$1"**: Fetches JSON data from the given URL silently.
* **jq .**: Formats (pretty-prints) the JSON output for readability.

**100. Run Unit Tests and Generate a Coverage Report (Python Example)**

bash

CopyEdit

#!/bin/bash

pytest --maxfail=1 --disable-warnings -q

coverage run -m pytest

coverage report

* **pytest --maxfail=1 --disable-warnings -q**: Runs Python unit tests using pytest, stopping on the first failure, and suppressing warnings for a cleaner output.
* **coverage run -m pytest**: Executes tests under the coverage tool to measure code coverage.
* **coverage report**: Generates a report summarizing the test coverage results.