Linux Production Shell Scripts

👉Mahesh Sarjerao Girhe [MSG] ✅

LinkedIn:-

https://www.linkedin.com/in/maheshgirhe7875

1. File Backup Script:

```
#!/bin/bash

backup_dir="/path/to/backup"
source_dir="/path/to/source"

# Create a timestamped backup of the source directory tar -
czf "$backup_dir/backup_$(date +%Y%m%d_%H%M%S).tar.gz"
"$source_dir"
```

2. System Monitoring Script:

```
#!/bin/bash threshold=90

# Monitor CPU usage and trigger alert if threshold exceeded
cpu_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -d. -
f1) if [ "$cpu_usage" -gt "$threshold" ]; then echo "High CPU usage
detected: $cpu_usage%"
    # Add alert/notification logic here
fi
```

3. User Account Management Script:

```
#!/bin/bash username="newuser"

# Check if user exists; if not, create new user
if id "$username" &>/dev/null; then echo "User
$username already exists."
else useradd -m "$username" echo
    "User $username created."
Fi
```

```
#
```

4. Log Analyzer Script:

```
#!/bin/bash

logfile="/path/to/logfile.log"

# Extract lines with "ERROR" from the log file
grep "ERROR" "$logfile" > error_log.txt echo
"Error log created."
```

5. Password Generator Script:

```
#!/bin/bash length=12

# Generate a random password
password=$(openssl rand -base64 $length)
echo "Generated password: $password"
```

6. File Encryption/Decryption Script:

```
#!/bin/bash file="/path/to/file.txt"

# Encrypt file using AES-256-CBC
openssl enc -aes-256-cbc -salt -in "$file" -out "$file.enc"
echo "File encrypted: $file.enc"
```

7. Automated Software Installation

Script:
```
#!/bin/bash packages=("package1"

"package2" "package3") Install listed

packages using apt-get for package in

"${packages[@]}"; do sudo apt-get

install "$package" -y
```

```
done echo "Packages installed

successfully."
```

8. Network Connectivity Checker Script:

```
#!/bin/bash host="example.com"

# Check network connectivity by pinging a host
if ping -c 1 "$host" &>/dev/null; then echo
"Network is up."
else echo "Network is
    down."
fi
```

9. Website Uptime Checker Script:

```
#!/bin/bash

website="https://example.com"

# Check if website is accessible
if curl --output /dev/null --silent --head --fail "$website"; then
    echo "Website is up."
else echo "Website is
    down."
fi
```

10. Data Cleanup Script: `#!/bin/bash`

```
directory="/path/to/cleanup"
```

```
#

  Remove files older than 7 days in specified directory
find "$directory" -type f -mtime +7 -exec rm {} \; echo
"Old files removed."
```

11. CPU Usage Tracker Script:

```
#!/bin/bash

output_file="cpu_usage_log.txt"

# Log current CPU usage to a file with timestamp echo "$(date) $(top
-bn1 | grep 'Cpu(s)' | awk '{print $2}' | cut -d.
-f1)%" >> "$output_file"
echo "CPU usage logged."
```

12. System Information Script:

```
#!/bin/bash

output_file="system_info.txt"

# Gather system information and save to a file
echo "System Information:" > "$output_file"
echo "--------------------" >> "$output_file"
echo "Hostname: $(hostname)" >> "$output_file"
echo "OS: $(uname -a)" >> "$output_file" echo
"Memory: $(free -h)" >> "$output_file" echo
"Disk Space: $(df -h)" >> "$output_file" echo
"System info saved to $output_file."
```

13. Task Scheduler Script:

```
#!/bin/bash

scheduled_task="/path/to/your_script.sh"
schedule_time="0 2 * * *"
  Schedule a task using cron echo "$schedule_time
$scheduled_task" | crontab echo "Task scheduled
successfully."
```

14. Disk Space Monitoring Script:

```
#!/bin/bash threshold=90

# Monitor disk usage and trigger alert if threshold exceeded
disk_usage=$(df -h | grep "/dev/sda1" | awk '{print $5}' | cut -d%
-f1)
if [ "$disk_usage" -gt "$threshold" ]; then echo
    "High disk usage detected: $disk_usage%"
    # Add alert/notification logic here
fi
```

15. Remote Server Backup Script:

```
#!/bin/bash

source_dir="/path/to/source"
remote_server="user@remoteserver:/path/to/backup"

# Backup files/directories to a remote server using rsync
rsync -avz "$source_dir" "$remote_server"
echo "Files backed up to remote server."
```

```
#
```

16. Environment Setup Script:

```
#!/bin/bash

# Customize for your specific environment setup echo
"Setting up development environment..." # Install
necessary packages, configure settings, etc. echo
"Development environment set up successfully."
```

17. File Compression/Decompression Script:

```
#!/bin/bash

file_to_compress="/path/to/file.txt"

# Compress a file using gzip
gzip "$file_to_compress"
echo "File compressed: $file_to_compress.gz"
```

18. Database Backup Script:

```
#!/bin/bash

database_name="your_database"
output_file="database_backup_$(date +%Y%m%d).sql"

# Perform database backup using mysqldump
mysqldump -u username -ppassword "$database_name" > "$output_file"
echo "Database backup created: $output_file"
```

19. Git Repository Updater Script:

```
#!/bin/bash

git_repo="/path/to/your/repo"

# Update a Git repository cd
"$git_repo" git pull origin
master echo "Git repository
updated."
```

20. Directory Synchronization Script:

```bash
#!/bin/bash

source_dir="/path/to/source"
destination_dir="/path/to/destination"
```

```
# Synchronize directories using rsync rsync -
avz "$source_dir" "$destination_dir" echo
"Directories synchronized successfully."
```

21. Web Server Log Analyzer Script:

```
#!/bin/bash

log_file="/var/log/apache2/access.log"

# Analyze web server log to count unique IP addresses
awk '{print $1}' "$log_file" | sort | uniq -c | sort -
nr echo "Web server log analyzed."
```

22. System Health Check Script:

```
#!/bin/bash

output_file="system_health_check.txt"

# Perform system health check and save results to a file
echo "System Health Check:" > "$output_file" echo "----
------------------" >> "$output_file" echo "Uptime:
$(uptime)" >> "$output_file"
echo "Load Average: $(cat /proc/loadavg)" >> "$output_file"
echo "Memory Usage: $(free -m)" >> "$output_file" echo
"System health check results saved to $output_file."
```

23. Automated Database Cleanup Script:

```
#!/bin/bash

database_name="your_database"
days_to_keep=7

# Clean up old database backups older than specified days find
/path/to/database/backups -name "$database_name*.sql" -mtime
```

```
+"$days_to_keep" -exec rm {} \; echo
"Old database backups cleaned up."
```

24. User Password Expiry Checker Script:

```
#!/bin/bash

# Check password expiry for users with bash shell
IFS=$'\n'
for user in $(cat /etc/passwd | grep "/bin/bash" | cut -d: -f1); do
    password_expires=$(chage -l "$user" | grep "Password expires" |
awk '{print $4}') echo "User: $user, Password Expires:
    $password_expires"
done
unset IFS
```

25. Service Restart Script:

```
#!/bin/bash

service_name="your_service"

# Restart a specified service sudo
systemctl restart "$service_name" echo
"Service $service_name restarted."
```

26. Folder Size Checker Script:

```
#!/bin/bash

folder_path="/path/to/folder"

# Check and display the size of a specified folder
du -sh "$folder_path" echo
"Folder size checked."
```

27. Backup Rotation Script:

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

```
backup_dir="/path/to/backups"

max_backups=5


# Rotate backups by deleting the oldest if more than max_backups
while [ $(ls -1 "$backup_dir" | wc -l) -gt "$max_backups" ]; do
oldest_backup=$(ls -1t "$backup_dir" | tail -n 1)
    rm -r "$backup_dir/$oldest_backup"
done
echo "Backup rotation completed."
```

28. Remote Script Execution Script:

```
#!/bin/bash

remote_server="user@remote-server"
remote_script="/path/to/remote/script.sh"

# Execute a script on a remote server via SSH ssh
"$remote_server" "bash -s" < "$remote_script"
echo "Remote script executed."
```

29. Network Interface Information Script:

```
#!/bin/bash network_interface="eth0"

# Display network interface information
ifconfig "$network_interface"
echo "Network interface information displayed."
```

30. Random Quotes Generator Script: `#!/bin/bash`

```
quotes=("Quote 1" "Quote 2" "Quote 3" "Quote 4")

# Generate and display a random quote from the array
random_index=$((RANDOM % ${#quotes[@]})) echo
"Random Quote: ${quotes[$random_index]}"
```