Below is a list of common scenarios where shell scripting can save the day! Whether you're automating repetitive tasks, monitoring system health, or managing files and processes, these scripts are designed to make your life easier. From checking network connectivity to generating SSL certificates, each script tackles a specific problem with simplicity and efficiency

1. **Monitor Network Connectivity**

```
#!/bin/bash
HOST="8.8.8.8"
LOGFILE="/var/log/network_monitor.log"
if ! ping -c 3 "$HOST" > /dev/null 2>&1; then
    echo "$(date): Server $HOST is unreachable" >> "$LOGFILE"
fi
```

-**Purpose**: Monitors network connectivity by pinging a host (e.g., Google's DNS server `8.8.8.8`).
-**How it works**:
- Sends 3 ICMP packets to the host.
- If the host is unreachable, logs the event with a timestamp to `/var/log/network_monitor.log`.

---

2. **Check Free Memory**

```
#!/bin/bash
THRESHOLD=10
FREE_MEM=$(free | awk '/Mem/{print $4/$2 * 100.0}')
if (( $(echo "$FREE_MEM < $THRESHOLD" | bc -l) )); then
    echo "Warning: Free memory below $THRESHOLD%" | mail -s "Low Memory Alert"
user@example.com
fi
```

-**Purpose**: Checks if free memory is below a specified threshold.
-**How it works**:
 - Calculates free memory as a percentage of total memory.
 - If free memory is below the threshold (e.g., 10%), sends an email alert.

---

3. **Monitor and Restart Processes**

```
#!/bin/bash
PROCESSES=(nginx mysql)
for PROC in "${PROCESSES[@]}"; do
    if ! pgrep -x "$PROC" > /dev/null; then
        echo "Restarting $PROC"
        systemctl restart "$PROC"
    fi
done
```

-**Purpose**: Monitors and restarts specified processes if they are not running.
-**How it works**:
 - Checks if processes like `nginx` and `mysql` are running using `pgrep`.
 - If a process is not running, restarts it using `systemctl`.

---

4. **Download Latest Backup**

```
#!/bin/bash
REMOTE_SERVER="user@remote:/backups/latest.tar.gz"
LOCAL_DIR="/local_backup"
scp "$REMOTE_SERVER" "$LOCAL_DIR" && echo "$(date): Backup downloaded" >>
backup.log
```

-**Purpose**: Downloads the latest backup from a remote server.
-**How it works**:
  - Uses `scp` to copy a file from the remote server to a local directory.
  - Logs the download event to `backup.log`.

---

5. **Create a New User**

```
#!/bin/bash
USERNAME="$1"
useradd -m -s /bin/bash "$USERNAME" && passwd "$USERNAME"
usermod -aG sudo "$USERNAME"
```

-**Purpose**: Creates a new user with sudo privileges.
-**How it works**:
  - Adds a new user with a home directory and bash shell.
  - Sets a password and adds the user to the `sudo` group.

---

6. **Move Large Files**

```
#!/bin/bash
SOURCE_DIR="/data"
DEST_DIR="/large_files"
mv $(find "$SOURCE_DIR" -type f -size +1G) "$DEST_DIR"
```

-**Purpose**: Moves files larger than 1GB from one directory to another.
-**How it works**:
  - Uses `find` to locate files larger than 1GB in the source directory.
  - Moves them to the destination directory.

---

7. **Check Uptime**

```
#!/bin/bash
UPTIME=$(awk '{print $1/86400}' /proc/uptime)
if (( $(echo "$UPTIME < 1" | bc -l) )); then
    echo "$(date): Uptime is less than 24 hours" >> uptime.log
fi
```

-**Purpose**: Checks if the system has been up for less than 24 hours.
-**How it works**:
- Reads the system uptime from `/proc/uptime` and converts it to days.
- Logs a message if uptime is less than 1 day.

---

8. **Check Disk Space on Multiple Servers**

```
#!/bin/bash
SERVERS=(server1 server2 server3)
THRESHOLD=90
for SERVER in "${SERVERS[@]}"; do
    USAGE=$(ssh "$SERVER" df / | awk 'NR==2 {print $5}' | sed 's/%//')
    if [ "$USAGE" -gt "$THRESHOLD" ]; then
        echo "Disk space warning on $SERVER: $USAGE% used" | mail -s "Disk Alert"
admin@example.com
    fi
done
```

-**Purpose**: Monitors disk usage on multiple servers.
-**How it works**:
- Connects to each server via SSH and checks disk usage.
- Sends an email alert if usage exceeds the threshold (e.g., 90%).

---

9. **Fetch and Analyze Logs**

```
#!/bin/bash
scp user@remote:/var/log/app.log .
grep "ERROR" app.log | tee error_report.log
```

-**Purpose**: Fetches and analyzes logs for errors.
-**How it works**:
- Copies a log file from a remote server.
- Searches for lines containing "ERROR" and saves them to `error_report.log`.

---

10. **Monitor Web Application**

```
#!/bin/bash
URL="http://example.com"
if ! curl -s --head "$URL" | grep "200 OK" > /dev/null; then
   echo "Restarting web application"
   systemctl restart apache2
fi
```

-**Purpose**: Monitors a web application and restarts it if it's down.
-**How it works**:
 - Sends an HTTP HEAD request to the URL.
 - If the response is not "200 OK", restarts the `apache2` service.

---

11. **Count Log File Lines**

```
#!/bin/bash
echo "Total lines: $(wc -l *.log | tail -n 1)"
```

-**Purpose**: Counts the total number of lines in all `.log` files.
-**How it works**:
 - Uses `wc -l` to count lines and `tail -n 1` to display the total.

---

12. **Compare Directories**

```
#!/bin/bash
diff -qr /dir1 /dir2
```

-**Purpose**: Compares two directories for differences.
-**How it works**:
 - Uses `diff` to show differences in files between `/dir1` and `/dir2`.

---

13. **Remove Old Logs**

```
#!/bin/bash
find /var/log -name "*.log" -mtime +7 -delete
```

-**Purpose**: Deletes log files older than 7 days.
-**How it works**:
  - Uses `find` to locate and delete `.log` files modified more than 7 days ago.

---

14. **Report Active Users**

```
#!/bin/bash
who | awk '{print $1}' | sort | uniq
```

-**Purpose**: Lists unique active users on the system.
-**How it works**:
  - Uses `who` to list users, `awk` to extract usernames, and `sort | uniq` to remove duplicates.

---

15. **Monitor Log File Size**

```
#!/bin/bash
LOG_DIR="/var/log"
THRESHOLD=500M
find "$LOG_DIR" -size +$THRESHOLD -exec ls -lh {} \;
```

-**Purpose**: Finds and lists log files larger than 500MB.
-**How it works**:
  - Uses `find` to locate files larger than the threshold and displays their sizes.

---

16. **Update Packages**

```
#!/bin/bash
apt update && apt upgrade -y && reboot
```

-**Purpose**: Updates system packages and reboots.
-**How it works**:
  - Runs `apt update` and `apt upgrade` to update packages, then reboots the system.

---

17. **Rotate Logs**

```
#!/bin/bash
LOG_DIR="/var/log"
tar -czf "$LOG_DIR/old_logs_$(date +%F).tar.gz" "$LOG_DIR"/*.log && rm "$LOG_DIR"/*.log
```

-**Purpose**: Archives and deletes old log files.
-**How it works**:
  - Compresses `.log` files into a tarball with a timestamp and deletes the original logs.

---

18. **Validate Checksums**

```
#!/bin/bash
find /files -type f -exec md5sum {} \; > checksums.md5
md5sum -c checksums.md5
```

-**Purpose**: Validates file integrity using checksums.
-**How it works**:
  - Generates MD5 checksums for files and verifies them.

---

19. **Install Missing Software**

```
#!/bin/bash
REQUIRED=(docker git)
for APP in "${REQUIRED[@]}"; do
    if ! command -v "$APP" > /dev/null; then
        apt install -y "$APP"
    fi
done
```

-**Purpose**: Installs missing software packages.
-**How it works**:
  - Checks if required applications (e.g., `docker`, `git`) are installed and installs them if missing.

---

20. **Generate SSL Certificate**

```
#!/bin/bash
openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout cert.key -out cert.pem -subj
"/CN=example.com"
```

-**Purpose**: Generates a self-signed SSL certificate.
-**How it works**:
  - Uses `openssl` to create a certificate valid for 365 days.

These scripts are versatile and can be adapted for various system administration and automation tasks. Let me know if you need further clarification!