Introduction to Shell Scripting

What is a Shell Script?

A shell script is a text file containing a sequence of commands for a Unix-based operating system's shell to execute. It can automate repetitive tasks, manage system administration, perform batch processing, and more.

Common Shells

- **Bash (Bourne Again Shell):** Most widely used.
- **sh (Bourne Shell):** The original Unix shell.
- **csh (C Shell):** Syntax resembles the C programming language.
- **ksh (Korn Shell):** Combines features of the Bourne and C Shells.
- **zsh (Z Shell):** Includes many features, like improved scripting and customization.

Basic Structure of a Shell Script

1. **Shebang Line: Indicates which shell should interpret the script.**

```
sh
```

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

Basic Structure of a Shell Script

2. Comments: Lines starting with # are comments

```sh
# This is a comment
```

Basic Structure of a Shell Script

3. Commands: Shell commands to be executed

```sh
echo "Hello, World!"
```

Running a Shell Script

1. Make the script executable:

```
sh

chmod +x script.sh
```

Running a Shell Script

2. Run the script:

```
sh
```

```
./script.sh
```

Examples and Concepts

Example 1: Hello World

sh

```sh
#!/bin/bash
# This script prints "Hello, World!" to the terminal.
echo "Hello, World!"
```

Variables

Variables store data that can be used and manipulated throughout the script.

```sh
#!/bin/bash
# Defining variables
NAME="Alice"
echo "Hello, $NAME"
```

User Input

You can prompt the user for input and use it in the script.

```sh
#!/bin/bash
# Prompting user for input
echo "Enter your name:"
read NAME
echo "Hello, $NAME"
```

Conditional Statements

Control the flow of the script based on conditions.

```sh
#!/bin/bash
# If-else statement
echo "Enter a number:"
read NUMBER

if [ "$NUMBER" -gt 10 ]; then
    echo "The number is greater than 10."
else
    echo "The number is 10 or less."
fi
```

Loops

Execute a block of code multiple times.

sh

```sh
#!/bin/bash
# For loop
for i in 1 2 3 4 5
do
    echo "Iteration number $i"
done
```

Functions

Group commands into reusable units.

```sh
#!/bin/bash
# Defining a function
greet() {
    echo "Hello, $1!"
}

# Calling the function
greet "Alice"
```

Example 2: Backup Script

This script backs up a directory to a specified location.

```sh
#!/bin/bash
# Backup script

# Source and destination directories
SOURCE_DIR="/path/to/source"
DEST_DIR="/path/to/destination"

# Check if source directory exists
if [ ! -d "$SOURCE_DIR" ]; then
   echo "Source directory does not exist."
   exit 1
fi

# Create backup
tar -czf "$DEST_DIR/backup_$(date +%F).tar.gz" -C "$SOURCE_DIR" .

echo "Backup completed successfully."
```

Example 3: System Monitoring Script

This script monitors system resource usage.

```sh
#!/bin/bash
# System monitoring script

# Check CPU usage
CPU_USAGE=$(top -b -n1 | grep "Cpu(s)" | awk '{print $2 + $4}')
echo "CPU Usage: $CPU_USAGE%"

# Check memory usage
MEMORY_USAGE=$(free | grep Mem | awk '{print $3/$2 * 100.0}')
echo "Memory Usage: $MEMORY_USAGE%"

# Check disk usage
DISK_USAGE=$(df -h | grep 'Filesystem\|/dev/sda1' | awk '{print $5}')
echo "Disk Usage: $DISK_USAGE"
```

Example 4: File Operations Script

This script performs various file operations.

sh

```sh
#!/bin/bash
# File operations script

# Creating a new directory
mkdir -p /tmp/mydir

# Creating a new file
echo "Hello, World!" > /tmp/mydir/hello.txt

# Appending to a file
echo "This is a new line." >> /tmp/mydir/hello.txt

# Reading a file
cat /tmp/mydir/hello.txt

# Deleting a file
rm /tmp/mydir/hello.txt

# Deleting the directory
rmdir /tmp/mydir
```