**Shell Scripting**

Shell scripting is a powerful tool for automating repetitive tasks and executing commands in Unix-like operating systems. Here are the basics:



* **Shell:** The shell is a command-line interpreter that takes commands from the keyboard and gives them to the operating system to perform. Common Unix shells include Bash (Bourne Again Shell), sh (Bourne Shell), csh (C Shell), and ksh (Korn Shell). Bash is the most common and widely used.



* **Script:** A shell script is a text file containing a sequence of shell commands that can be executed by the shell. Shell scripts typically have the .sh extension.



* **Shebang:** The first line of a shell script is called the shebang, which specifies the shell to be used to execute the script. For Bash scripts, the shebang line is #!/bin/bash.



* **Comments:** Comments in shell scripts start with the # symbol and are ignored by the shell. They are used to document the script's purpose and explain the code.



* **Variables:** Variables in shell scripts are defined without specifying a data type and are accessed using the $ prefix. For example:



NAME="John"

echo "Hello, $NAME!"

Command substitution: Command substitution allows you to capture the output of a command and store it in a variable. It is done using $(...) or backticks `...`. For example:

DATE=$(date)

echo "Today's date is $DATE"

**Conditional statements:** Conditional statements allow you to execute certain commands based on the result of a condition. Common constructs include if, elif, and else. For example:



#!/bin/bash



if [ $1 -gt 10 ]; then



echo "$1 is Greater than 10"



else



echo "$1 is Less than 10"



fi



User input:

#!/bin/bash

# Take input from the user

read -p "Enter a number: " num

# Check if the number is less than 5

if [ "$num" -lt 5 ]; then

echo "The number is less than 5."

else

echo "The number is 5 or greater."

fi

Loops: Loops allow you to execute a block of code repeatedly. Common loop constructs include for and while. For example:

for i in {1..5}; do

echo "Number: $i"

done

#!/bin/bash  
  
echo "enter a number:"  
read number  
  
for ((i=1; i<=number; i++))  
do  
  echo $i  
done

**Functions:**

Functions allow you to group commands together and call them by name. They are defined using the function keyword or simply by naming the function followed by parentheses (). For example:

#!/bin/bash

greet() {

echo "Hello, $1!"

}

greet "Alice"

#!/bin/bash

# Function to calculate factorial with parameter

factorial() {

local num=$1

local fact=1

# Check if the input is a non-negative number

if [ "$num" -lt 0 ]; then

echo "Factorial is not defined for negative numbers!"

exit 1

fi

# Calculate factorial using a loop

for (( i=1; i<=num; i++ ))

do

fact=$((fact \* i))

done

echo "$fact"

}

# Check if a parameter is provided

if [ -z "$1" ]; then

read -p "Enter a number: " num

else

num=$1

fi

# Call the factorial function with parameter

result=$(factorial "$num")

echo "Factorial of $num is: $result"

**File handling:**

Shell scripting provides various commands for file handling, such as creating, reading, writing, and deleting files. Common commands include touch, cat, grep, sed, awk, rm, mv, and cp.

These are just the basics of shell scripting, and there's much more you can do with it, including error handling, input/output redirection, and advanced scripting techniques.

**file\_manager.sh**

#!/bin/bash

# Function to create a file

create\_file() {

read -p "Enter the filename to create: " filename

touch "$filename"

echo "File '$filename' created successfully!"

}

# Function to copy a file

copy\_file() {

read -p "Enter the source file: " src

read -p "Enter the destination: " dest

if [ -f "$src" ]; then

cp "$src" "$dest"

echo "File '$src' copied to '$dest' successfully!"

else

echo "Source file '$src' does not exist!"

fi

}

# Function to move a file

move\_file() {

read -p "Enter the source file: " src

read -p "Enter the destination: " dest

if [ -f "$src" ]; then

mv "$src" "$dest"

echo "File '$src' moved to '$dest' successfully!"

else

echo "Source file '$src' does not exist!"

fi

}

# Function to delete a file

delete\_file() {

read -p "Enter the filename to delete: " filename

if [ -f "$filename" ]; then

rm "$filename"

echo "File '$filename' deleted successfully!"

else

echo "File '$filename' does not exist!"

fi

}

# Function to list files

list\_files() {

ls -lah

}

# Display menu options

while true; do

echo "Choose an option:"

echo "1) Create a file"

echo "2) Copy a file"

echo "3) Move a file"

echo "4) Delete a file"

echo "5) List files"

echo "6) Exit"

read -p "Enter your choice (1-6): " choice

case $choice in

1) create\_file ;;

2) copy\_file ;;

3) move\_file ;;

4) delete\_file ;;

5) list\_files ;;

6) echo "Exiting..."; exit 0 ;;

\*) echo "Invalid choice! Please enter a number between 1 and 6." ;;

esac

done