Lab 1: Basics of Shell Scripting

July 28, 2025

Objective

This lab introduces students to foundational concepts of shell scripting in Linux. It covers creating and executing basic shell scripts, working with variables, using operators, implementing control flow through loops, and writing simple functions. The goal is to help students gain hands-on experience with scripting constructs that are essential for automation and system programming tasks in Linux environments.

Shell Basics Scripting

Command Execution:

```
[202463010@paramshavak ~]$ nano shExample1.sh
[202463010@paramshavak ~]$ chmod 777 shExample1.sh
[202463010@paramshavak ~]$ ./shExample1.sh
What is your Name?
Shivam Solanki
Hello, Shivam Solanki
```

Script Content:

Listing 1: shExample1.sh

```
#!/bin/sh
# Author : Shivam Solanki

echo "What is your Name?"
read PERSON
echo "Hello, $PERSON"
```

Variables

Read-Only Variable

Command Execution:

```
[202463010@paramshavak ~]$ ./readonly_variable.sh
Permission denied
[202463010@paramshavak ~]$ chmod 777 readonly_variable.sh
[202463010@paramshavak ~]$ ./readonly_variable.sh
./readonly_variable.sh: line 9: NAME: readonly variable
```

Script Content:

Listing 2: readonly_variable.sh

```
#!/bin/sh

NAME="Shivam"
readonly NAME
NAME="Abhinav"
```

Unset Variable

Command Execution:

```
[202463010@paramshavak ~]$ ./unset_variable.sh
[202463010@paramshavak ~]$ chmod 777 unset_variable.sh
[202463010@paramshavak ~]$ ./unset_variable.sh

[202463010@paramshavak ~]$
```

Script Content:

Listing 3: unset_variable.sh

```
#!/bin/sh

NAME="Shivam"
unset NAME
echo $NAME
```

Special Variables

Command Execution:

```
[202463010@paramshavak ~]$ nano special_vars.sh
[202463010@paramshavak ~]$ chmod 777 special_vars.sh
[202463010@paramshavak ~]$ ./special_vars.sh Welcome to OS
File name: ./special_vars.sh
First parameter: Welcome
Third parameter: OS
Quoted Values ($@): Welcome to OS
Quoted Values ($*): Welcome to OS
Number of Parameters: 3
Exit status of last command: 0
Process ID (PID) of script: 223783
Current Date and Time using $(date): Mon Jul 28 14:14:59 IST 2025
```

Script Content:

Listing 4: special_vars.sh

```
#!/bin/sh

echo "File name: $0"
echo "First parameter: $1"
echo "Third parameter: $3"
echo "Quoted Values (\$@): $@"
echo "Quoted Values (\$*): $*"
echo "Number of Parameters: $#"
echo "Exit status of last command: $?"
echo "Process ID (PID) of script: $$"
current_date=$(date)
echo "Current Date and Time using \$(date): $current_date"
```

Basic Operators

Command Execution:

```
[202463010@paramshavak ~]$ nano basic_operators.sh
[202463010@paramshavak ~]$ chmod 777 basic_operators.sh
[202463010@paramshavak ~]$ ./basic_operators.sh
Values: a=20, b=10
Addition (a + b): 30
Subtraction (a - b): 10
Multiplication (a * b): 200
Division (a / b): 2
Modulus (a % b): 0
Assignment (c = b): Now c = 10
Equality Check (a == b): a is NOT equal to b
Not Equal Check (a != b): a is NOT equal to b
```

Script Content:

Listing 5: basic_operators.sh

```
#!/bin/sh
a=20 b=10
echo "Values: a=$a, b=$b"
echo "Addition (a + b):"
expr $a + $b
echo "Subtraction (a - b):"
expr $a - $b
echo "Multiplication (a * b):"
expr $a \* $b
echo "Division (a / b):"
expr $a / $b
echo "Modulus (a % b):"
expr $a % $b
echo "Assignment (c = b):"
c=$b
echo "Now c = $c"
echo "Equality Check (a == b):"
if [ $a -eq $b ]; then
  echo "a is equal to b"
else
  echo "a is NOT equal to b"
fi
echo "Not Equal Check (a != b):"
if [ $a -ne $b ]; then
  echo "a is NOT equal to b"
else
  echo "a is equal to b"
fi
```

Loops

For Loop

Command Execution:

```
[202463010@paramshavak ~]$ nano for_loop.sh
[202463010@paramshavak ~]$ chmod 777 for_loop.sh
[202463010@paramshavak ~]$ ./for_loop.sh
0
1
2
3
4
5
6
7
8
9
```

Script Content:

Listing 6: for_loop.sh

```
#!/bin/sh
# This loop will print numbers from 0 to 9

for var in 0 1 2 3 4 5 6 7 8 9
do
  echo $var
done
```

While Loop

Command Execution:

```
[202463010@paramshavak ~]$ nano while_loop.sh
[202463010@paramshavak ~]$ chmod 777 while_loop.sh
[202463010@paramshavak ~]$ ./while_loop.sh
0
1
2
3
4
5
6
7
8
9
```

Listing 7: while_loop.sh

```
#!/bin/sh
# This while loop prints numbers from 0 to 9

a=0
while [ $a -lt 10 ]
do
  echo $a
  a=`expr $a + 1`
done
```

While Loop with Break

Command Execution:

```
[202463010@paramshavak ~]$ nano while_break_loop.sh
[202463010@paramshavak ~]$ chmod 777 while_break_loop.sh
[202463010@paramshavak ~]$ ./while_break_loop.sh
0
1
2
3
4
5
```

Listing 8: while_break_loop.sh

```
#!/bin/sh
# This loop will stop once $a equals 5

a=0
while [ $a -lt 10 ]
do
  echo $a
  if [ $a -eq 5 ]
  then
    break
  fi
  a=`expr $a + 1`
done
```

Nested Loop

```
[202463010@paramshavak ~]$ nano nested_loop.sh
[202463010@paramshavak ~]$ chmod 777 nested_loop.sh
[202463010@paramshavak ~]$ ./nested_loop.sh
0
1 0
2 1 0
3 2 1 0
4 3 2 1 0
5 4 3 2 1 0
6 5 4 3 2 1 0
7 6 5 4 3 2 1 0
8 7 6 5 4 3 2 1 0
9 8 7 6 5 4 3 2 1 0
```

Listing 9: nested_loop.sh

```
#!/bin/sh
# Outer loop runs while a < 10
# Inner loop runs while b >= 0

a=0
while [ "$a" -lt 10 ]       # this is loop 1
do
  b="$a"
  while [ "$b" -ge 0 ]      # this is loop 2
  do
    echo -n "$b "
    b=‘expr $b - 1‘
  done
  echo                       # newline after each inner loop
  a=‘expr $a + 1‘
done
```

Functions

Function with Return

Command Execution:

```
[202463010@paramshavak ~]$ nano function_with_return.sh
[202463010@paramshavak ~]$ chmod 777 function_with_return.sh
[202463010@paramshavak ~]$ ./function_with_return.sh
Hello Shivam Solanki
Return value is 10
```

Script Content:

Listing 10: function_with_return.sh

```
#!/bin/sh
# Define function with parameters and return value

Hello() {
    echo "Hello $1 $3"
    return 10
}

# Call the function with three arguments
Hello Shivam Nitin Solanki

# Capture return value
ret=$?
echo "Return value is $ret"
```

Nested Functions

Command Execution:

```
[202463010@paramshavak ~]$ nano nested_function.sh
[202463010@paramshavak ~]$ chmod 777 nested_function.sh
[202463010@paramshavak ~]$ ./nested_function.sh
Shivam online...Over
Abhinav online...Over
```

Script Content:

Listing 11: nested_function.sh

```
#!/bin/sh
# Define function that calls another function

number_one() {
    echo "Shivam online...Over"
    number_two
}

number_two() {
    echo "Abhinav online...Over"
}

# Start by calling number_one()
number_one
```

Assignment – Shell Script Utility Toolkit

Objective

Design a shell script that gives users a **menu-based utility** to perform two common tasks:

1. Organize files by type in a given directory
2. Scan a subnet to find active hosts in the local network

The goal is to test student's ability to work with file operations, loops, functions, user input, and parallel process handling in sh.

Task Description

Your script should:

- Display a menu:

  ```
  ===== Shell Utility Toolkit =====
  1. Organize Files by Type
  2. Scan Network for Active Hosts
  3. Exit
  ```

- Based on user input:

  Option 1: Ask for a directory path. Organize all files based on their extension into subfolders (e.g., `.txt`, `.sh`). Files with no extension go into a `no_ext` folder.

  Option 2: Ask for a subnet prefix (e.g., `10.100.255.`). Ping each host from .1 to .254 using parallel execution. Display only live IPs.

- Use `functions`, `conditionals`, and proper `input validation`.

Sample Interaction

```
===== Shell Utility Toolkit =====
1. Organize Files by Type
2. Scan Network for Active Hosts
3. Exit
Enter your choice: 2

Enter subnet (e.g., 10.100.255.): 10.100.255.
Scanning...
Node with IP: 10.100.255.44 is up.
Node with IP: 10.100.255.71 is up.
```