CompTIA Linux+

Working with Bash Profiles and Bash Scripts

- **Introduction**
- **Lab Topology**
- **Exercise 1 - Working with Bash Profiles and Bash Scripts**
- **Review**

Introduction

Welcome to the Working with Bash Profiles and Bash Scripts Practice Lab. In this module you will be provided with the instructions and devices needed to develop your hands-on skills.

Bash Profiles
Bash Scripts
Parameters

Learning Outcomes

In this module, you will complete the following exercise:

- Exercise 1 - Working with Bash Profiles and Bash Scripts

After completing this lab, you will be able to:

- Understand the role of various bash related files
- Write a simple bash script
- Use commenting
- Use parameters
- Capture user inputs in scripts

Exam Objectives

The following exam objectives are covered in this lab:

- LPI: 105.1 Customize and use the shell environment
- CompTIA: 2.2 Given a scenario, manage users and groups.

Exercise 1 - Working with Bash Profiles and Bash Scripts

Bash is a shell, which acts as a command language interpreter. When a user executes a command, the output is generated as the output of the command. Bash can execute individual commands as well as the commands from a file, which is known as a script.

In this exercise, you will work with the Bash profiles and Bash scripts.

Learning Outcomes

After completing this exercise, you will be able to:

- Log into a Linux system
- Understanding the role of various bash related files
- Write a simple bash script
- Use commenting
- Use parameters
- Capture user inputs in scripts

Task 1 - Understand the Role of Various Bash Related Files

Bash contains two different kinds of files: system and user configuration files. The role of each type of files differs.

In this task, you will read the environment variables.

Step 1

On the desktop, right-click and select Open Terminal.

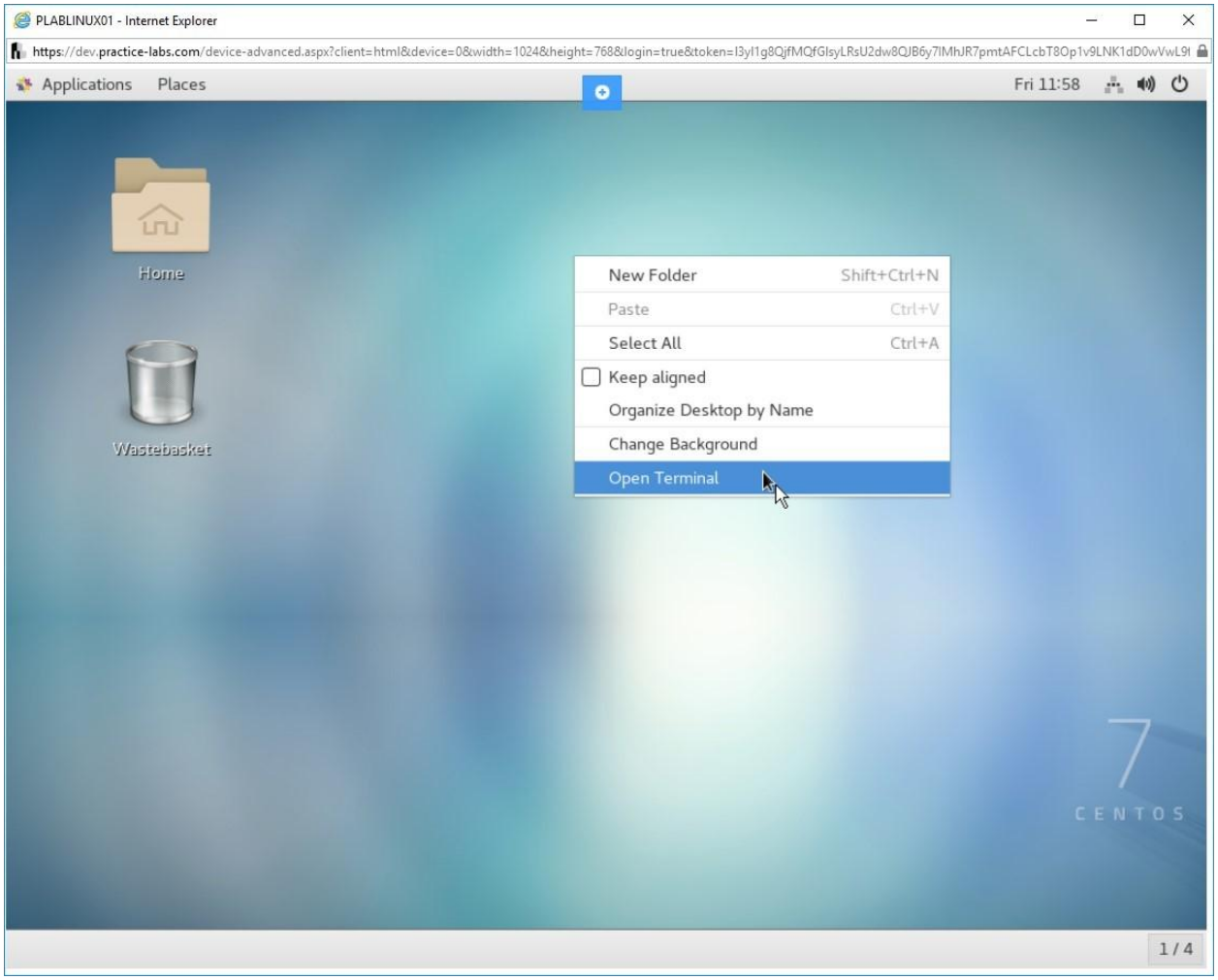


Figure 1.1 Screenshot of PLABLINUX01: Selecting the Open Terminal option from the context menu.

Step 2

The terminal window is displayed.

Bash reads the /etc/profile file for instructions if:

- Invoked interactively using the --login option
- Invoked as sh

The /etc/profile contains several shell variables, such as PATH, USER, MAIL, HOSTNAME, and HISTSIZE.

To view the /etc/profile file, type the following command:

```
cat /etc/profile
```

Press Enter.

Note: You should maximize the terminal window to view these files.

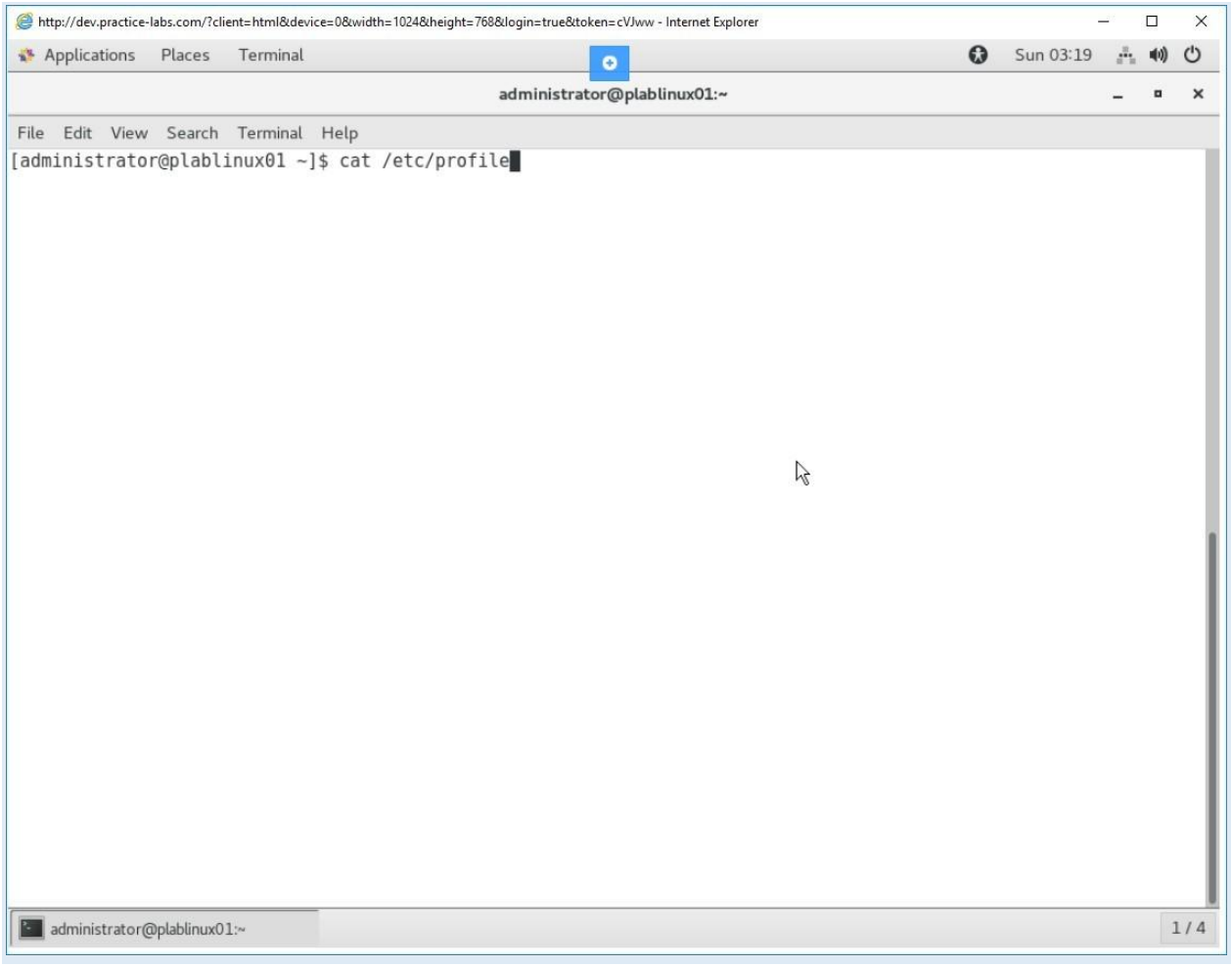


Figure 1.2 Screenshot of PLABLINUX01: Viewing the /etc/profile file.

Step 3

Notice various shell variables are already set in this profile. The /etc/profile file will contain the configuration settings that must be applied to all user environments.

Notice the highlighted shell variables.

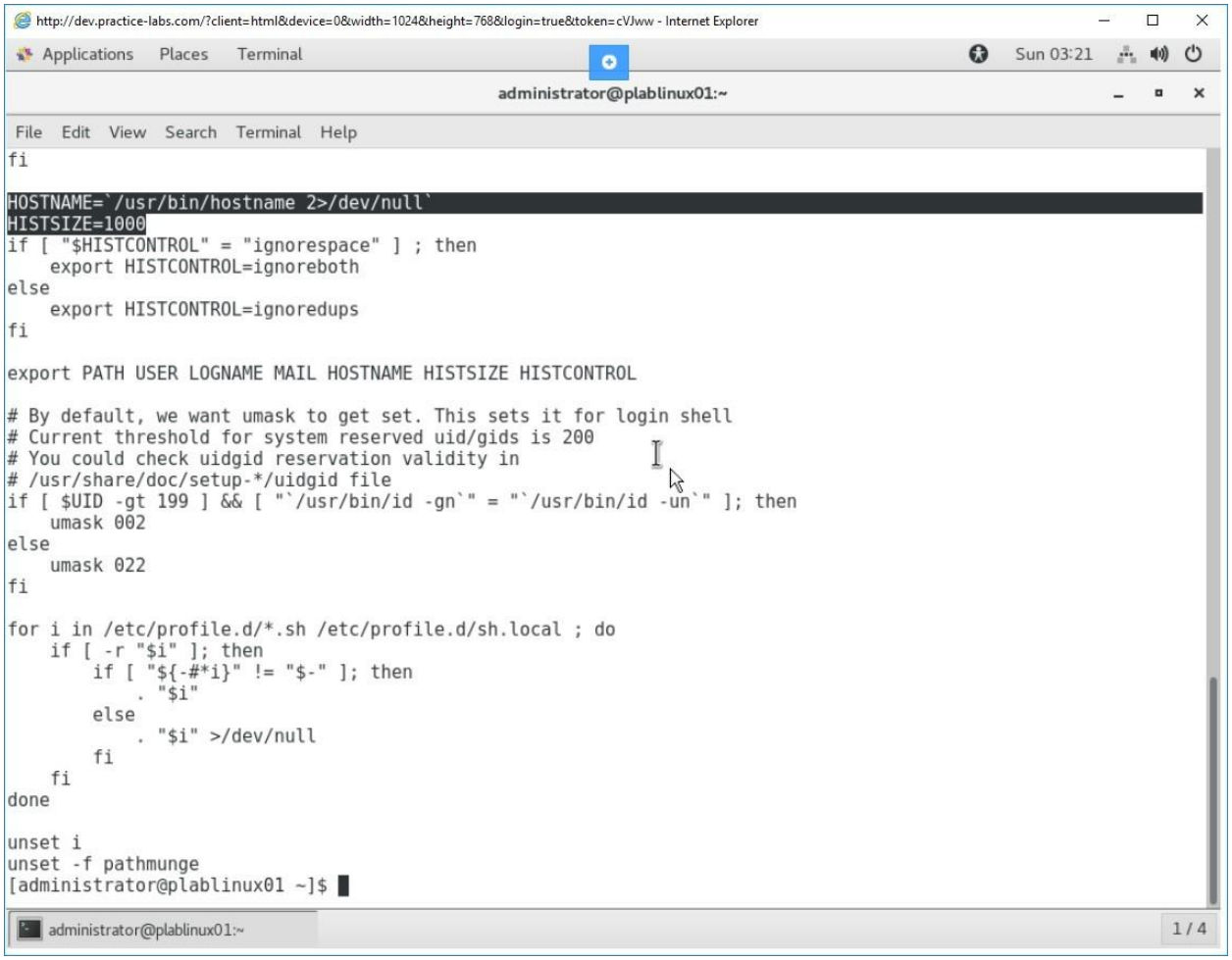


Figure 1.3 Screenshot of PLABLINUX01: Viewing the /etc/profile file.

Step 4

There may be cases when a Linux system contains more than one shell. The /etc/profile file, in this case, will be read by all shells existing on the system.

There can be configurations in the Linux environment in which the /etc/profile file contains only the shell environment and program startup settings.

On the other hand, the /etc/bashrc file contains system-wide definitions for shell functions and aliases.

Clear the screen by entering the following command:

```
clear
```

To view the /etc/bashrc file, type the following command:

```
cat /etc/bashrc
```

Press Enter.

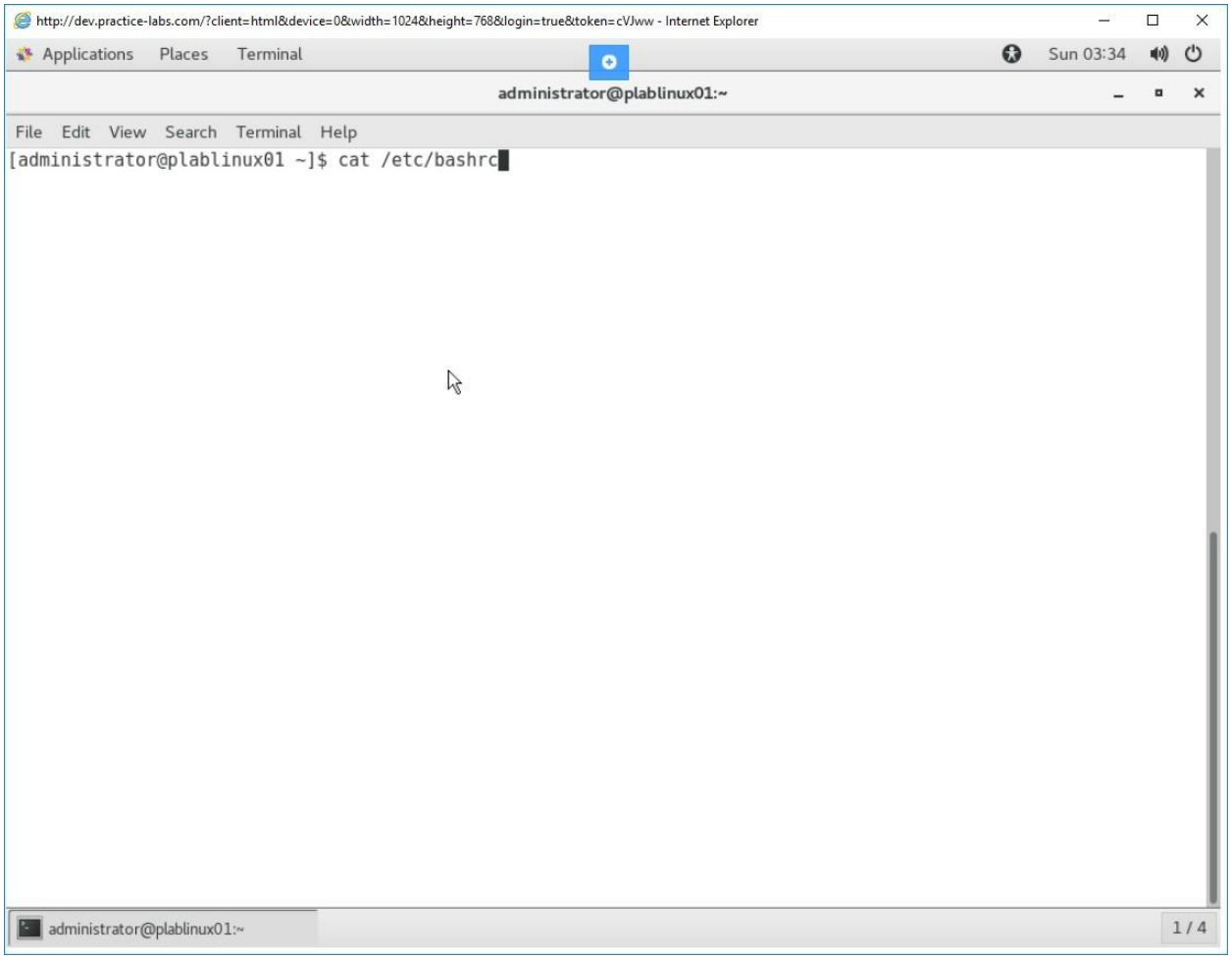


Figure 1.4 Screenshot of PLABLINUX01: Displaying the /etc/bashrc file.

Step 5

The output of the /etc/bashrc file is displayed.

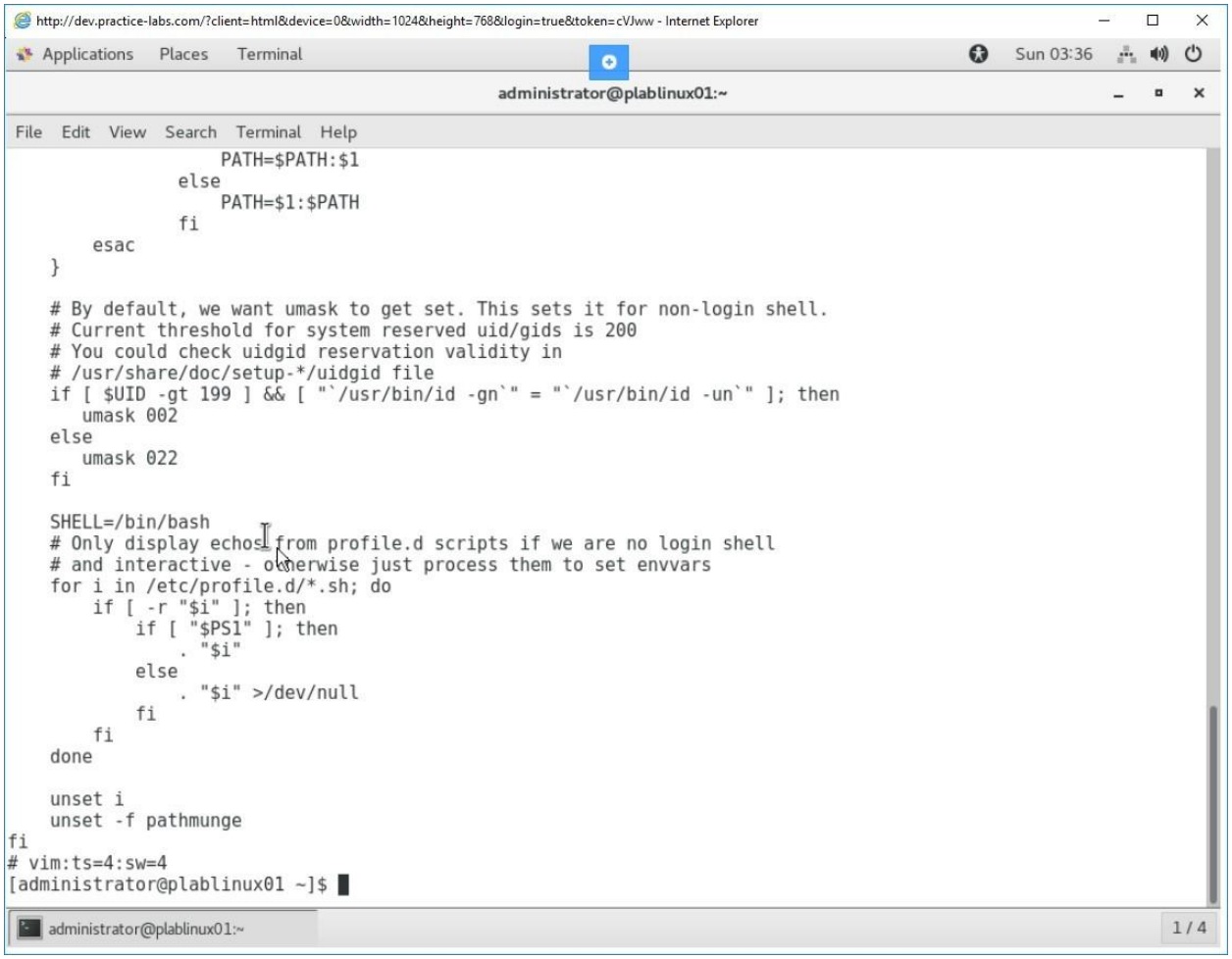


Figure 1.5 Screenshot of PLABLINUX01: Displaying the /etc/bashrc file.

Step 6

Notice the shell configuration setting.

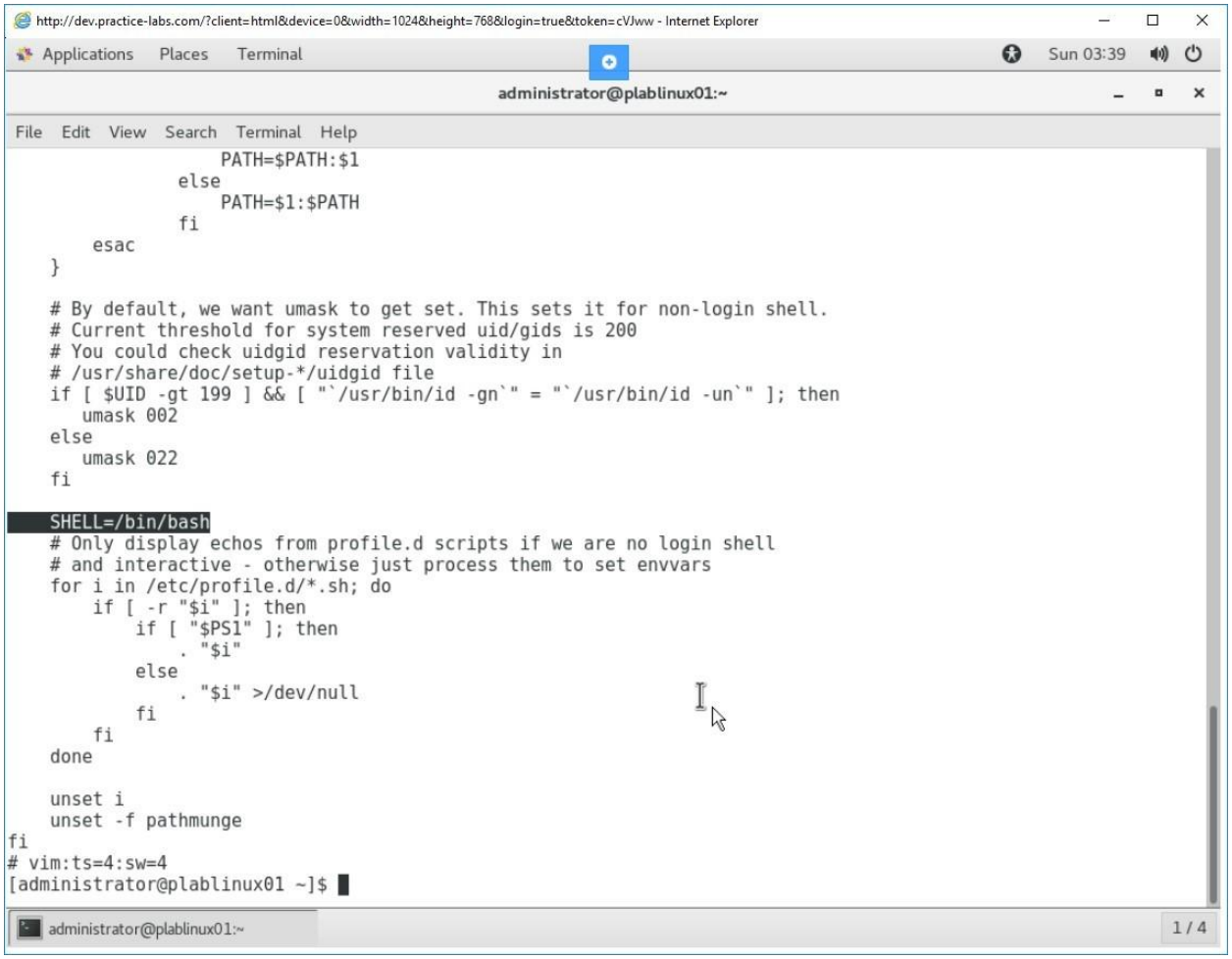


Figure 1.6 Screenshot of PLABLINUX01: Displaying the /etc/bashrc file.

Step 7

This file also contains the settings to be executed if you are using an interactive or non-interactive shell

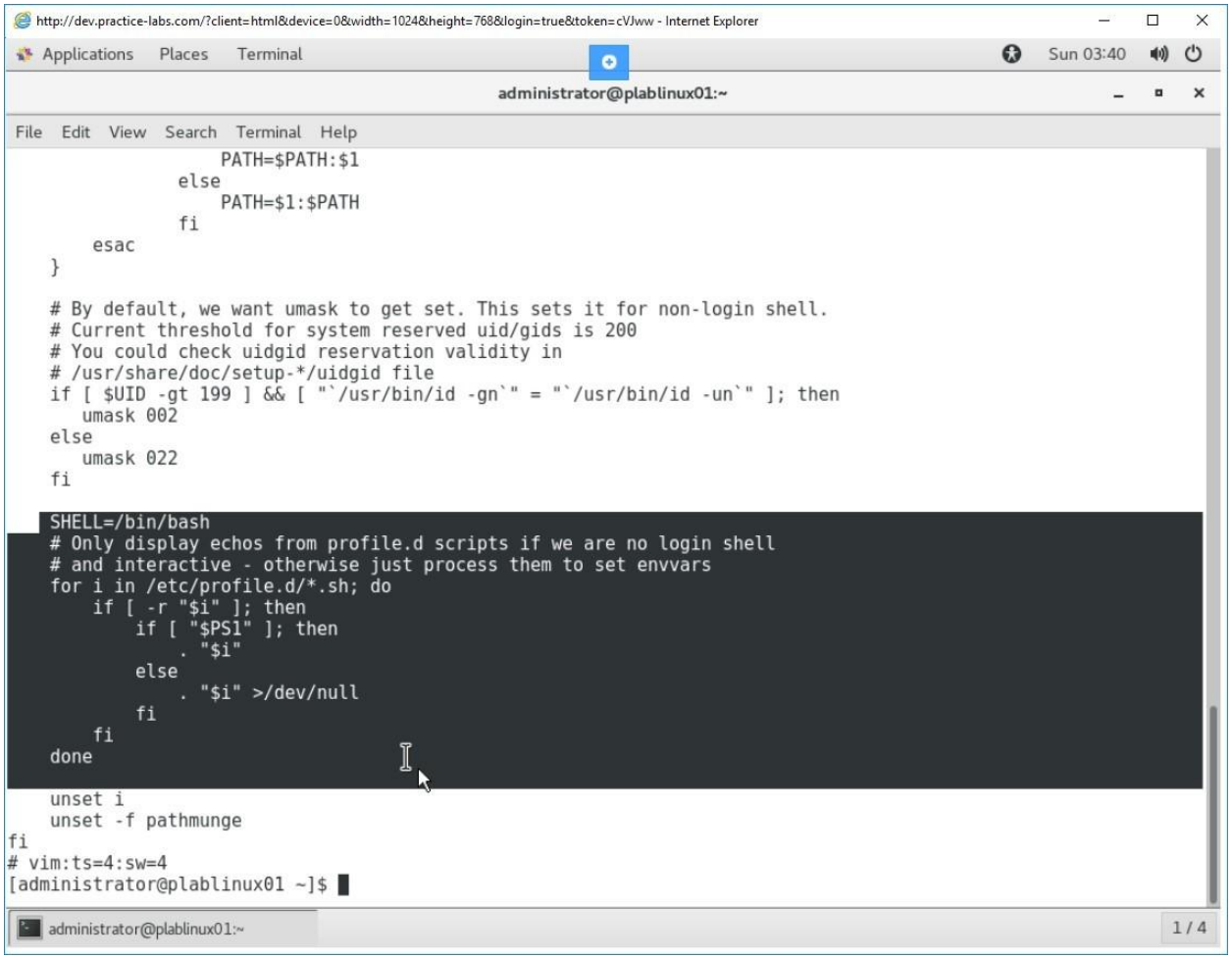


Figure 1.7 Screenshot of PLABLINUX01: Displaying the /etc/bashrc file.

Step 8

Clear the screen by entering the following command:

```
clear
```

Let's now view the user configuration files. The ~/.bash_profile is the user configuration file in which the user environment can be configured. By default, some configuration is already defined, but it can be changed or altered as per requirements.

To view the ~/.bash_profile file, type the following command:

```
cat ~/.bash_profile
```

Press Enter.

Note: If the ~/.bash_profile does not exist, then the ~/.bash_login file is read.

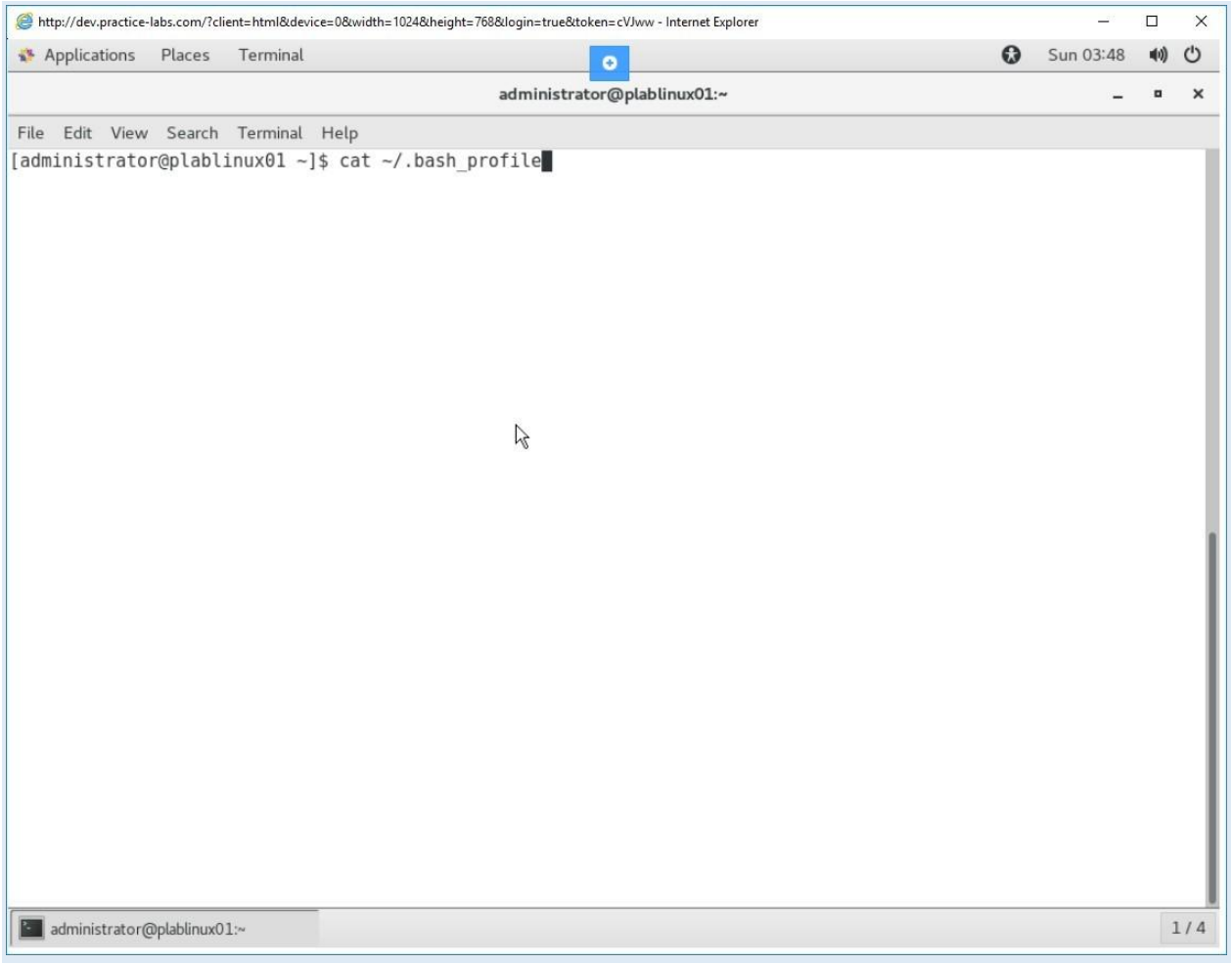


Figure 1.8 Screenshot of PLABLINUX01: Viewing the ~/.bash_profile file.

Step 9

The output of the file is displayed.

Note: The ~/.bash_profile gets executed only during a login shell. When Bash is invoked as a login shell, the order of file execution is as follows:

/etc/profile > ~/.bash_profile > ~/.bash_login >~/.profile. The bash_profile executes the bashrc file.

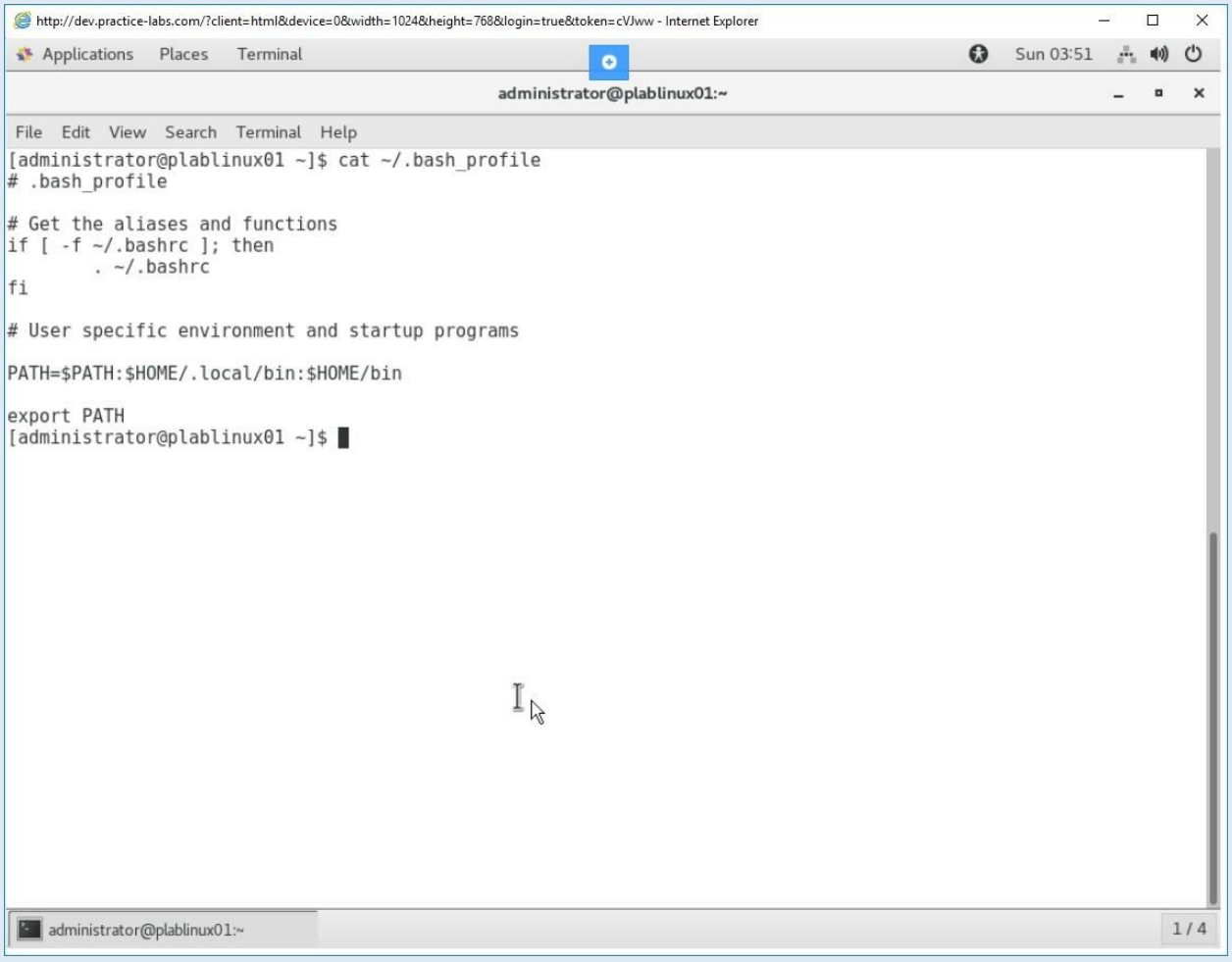


Figure 1.9 Screenshot of PLABLINUX01: Displaying the ~/.bash_profile file.

Step 10

Clear the screen by entering the following command:

```
clear
```

The ~/.bash_login file contains the settings that are executed when a user logs on to the Linux system.

To view the ~/.bash_login file, type the following command:

```
cat ~/.bash_login
```

Press Enter.

Notice that this file does not exist. However, it can be created. In the absence of this file, the ~/.profile file is read.

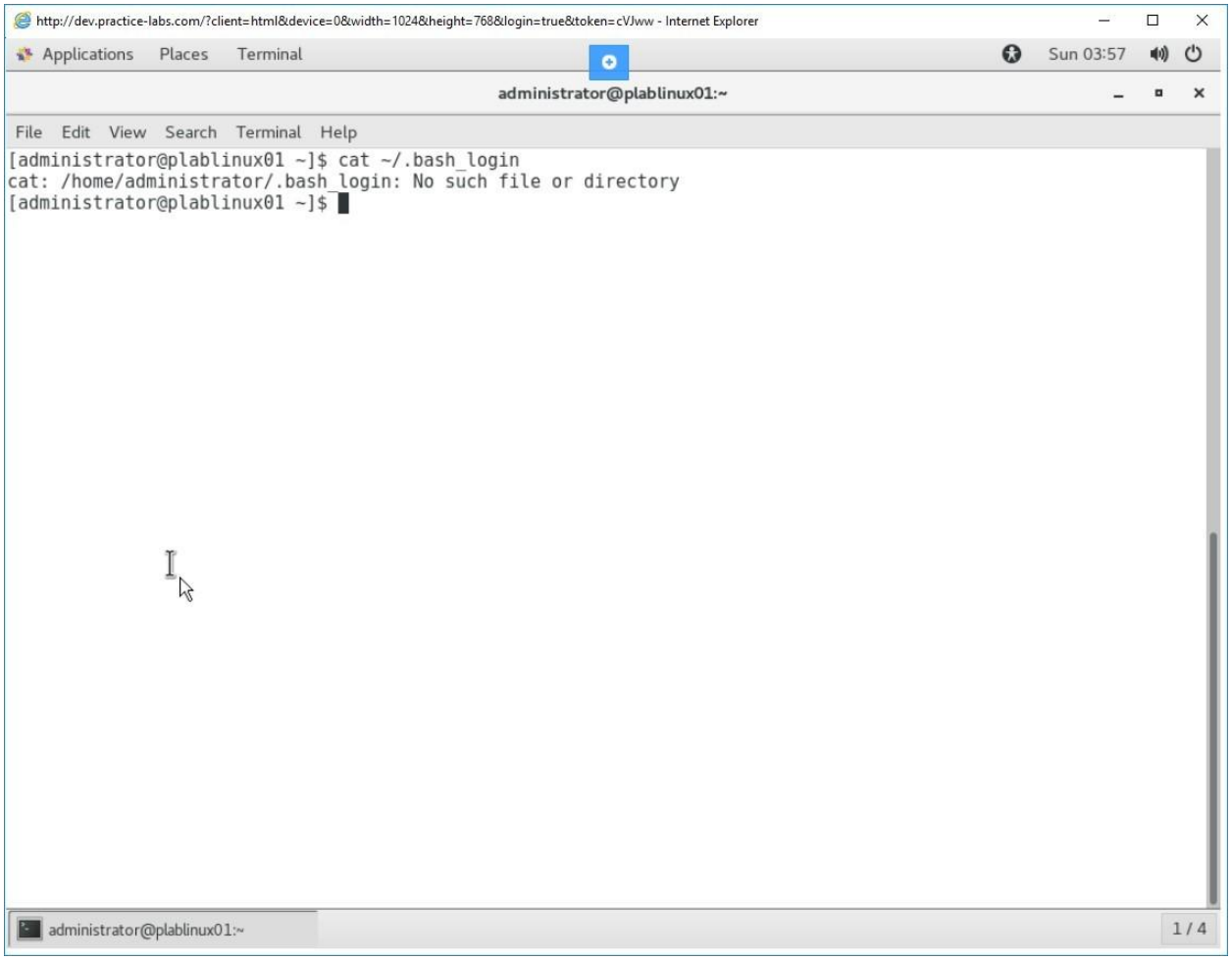


Figure 1.10 Screenshot of PLABLINUX01: Attempting to open the bash_login file.

Step 11

Clear the screen by entering the following command:

```
clear
```

The ~/.profile file contains the same settings as the ~/.bash_profile and ~/.bash_login files.

To view the ~/.profile file, type the following command:

```
cat ~/.profile
```

Press Enter.

Notice that this profile file does not exist. However, it can be created.

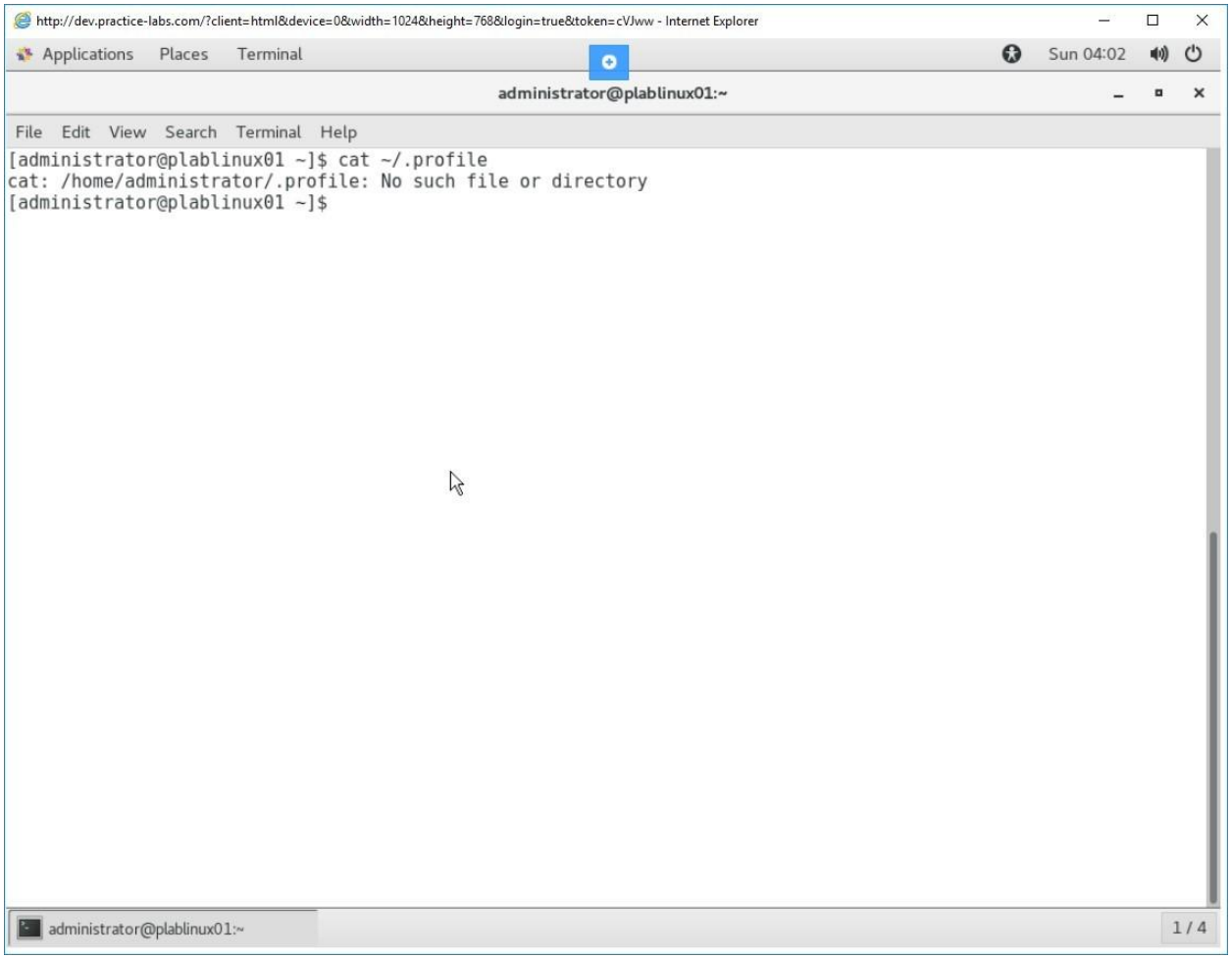


Figure 1.11 Screenshot of PLABLINUX01: Attempting to open the profile file.

Step 12

Clear the screen by entering the following command:

```
clear
```

The ~/.bashrc script executes the /etc/bashrc file. To view the ~/.bashrc file, type the following command:

```
cat ~/.bashrc
```

Press Enter.

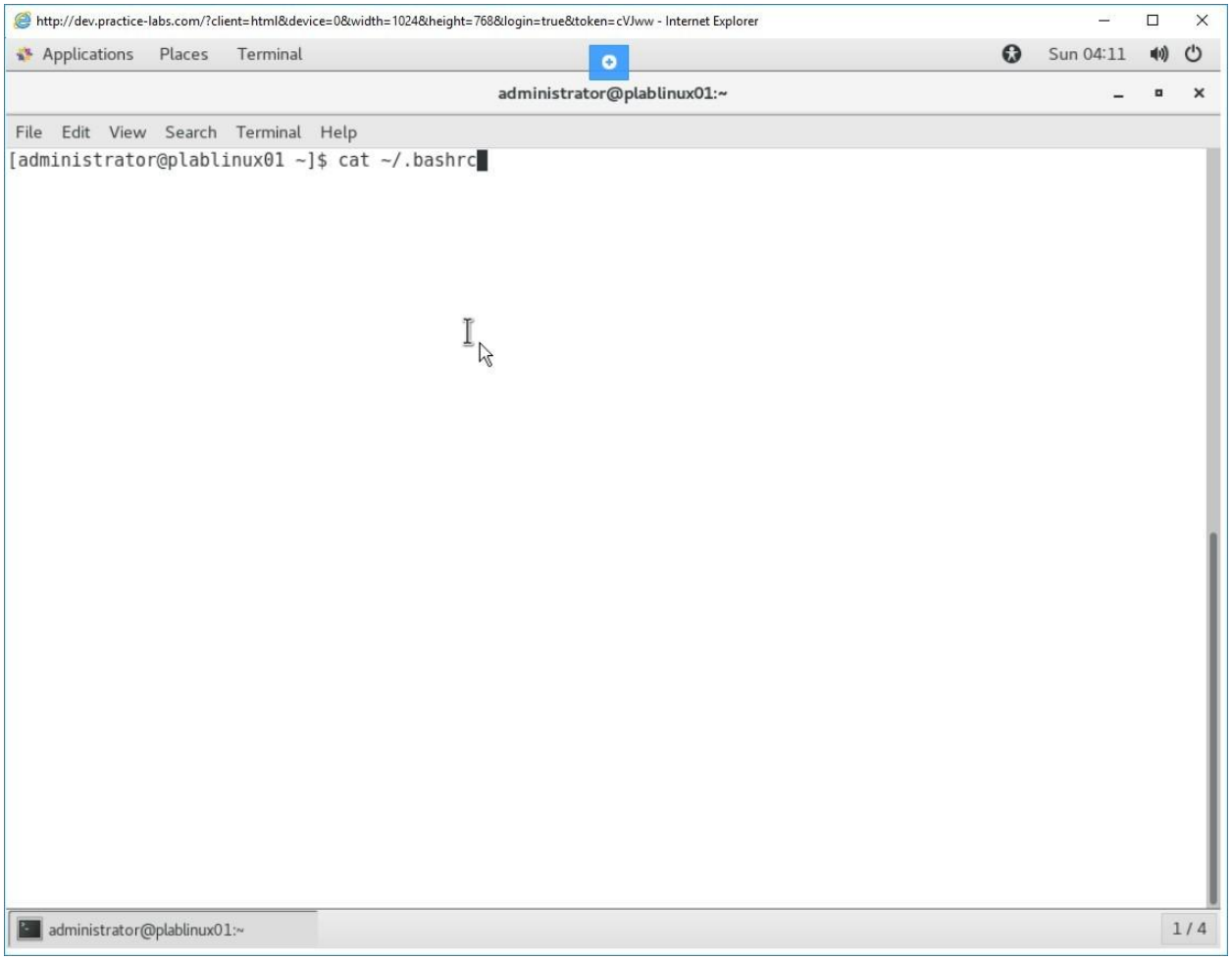


Figure 1.12 Screenshot of PLABLINUX01: Opening the bashrc file.

Step 13

The output is displayed.

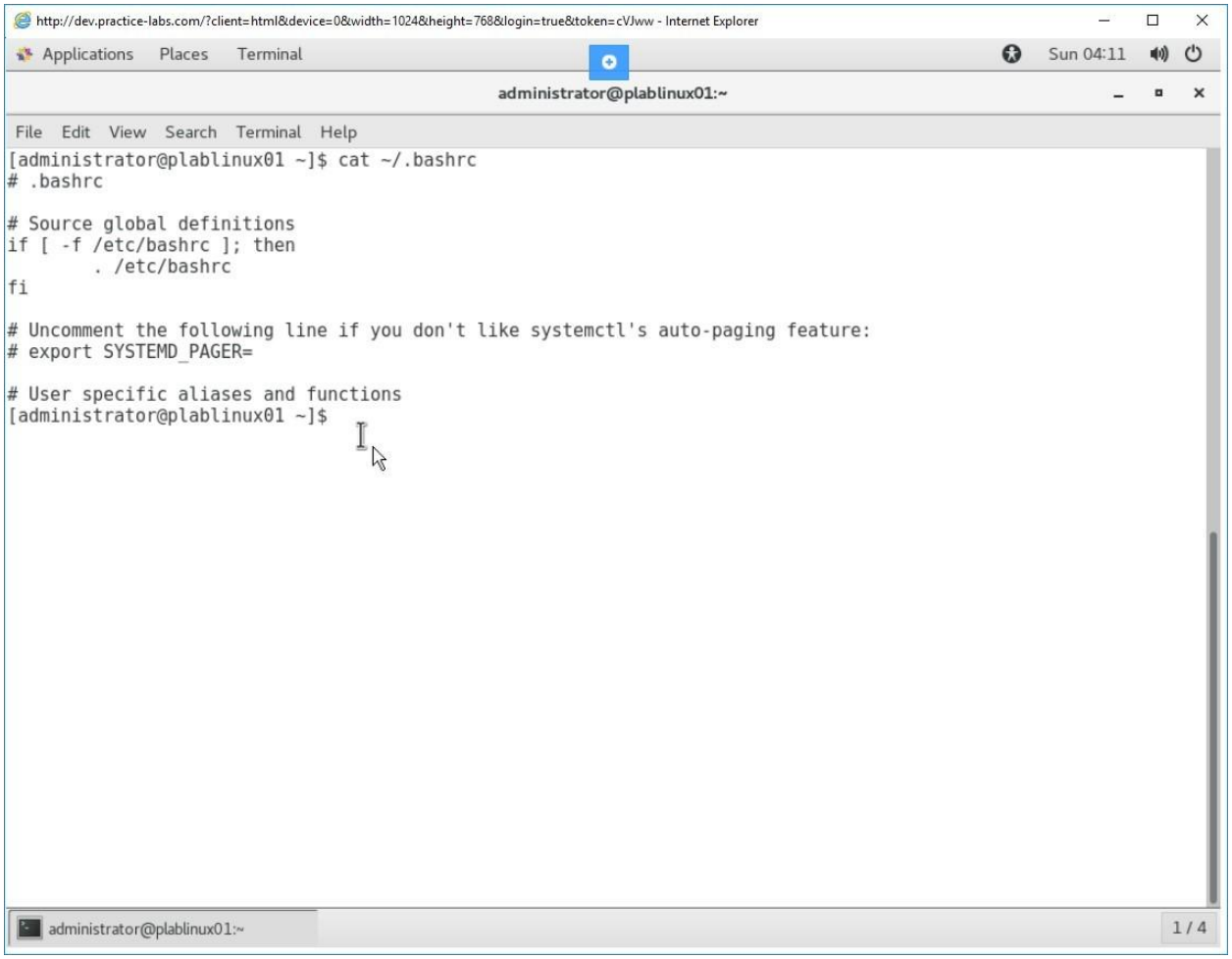


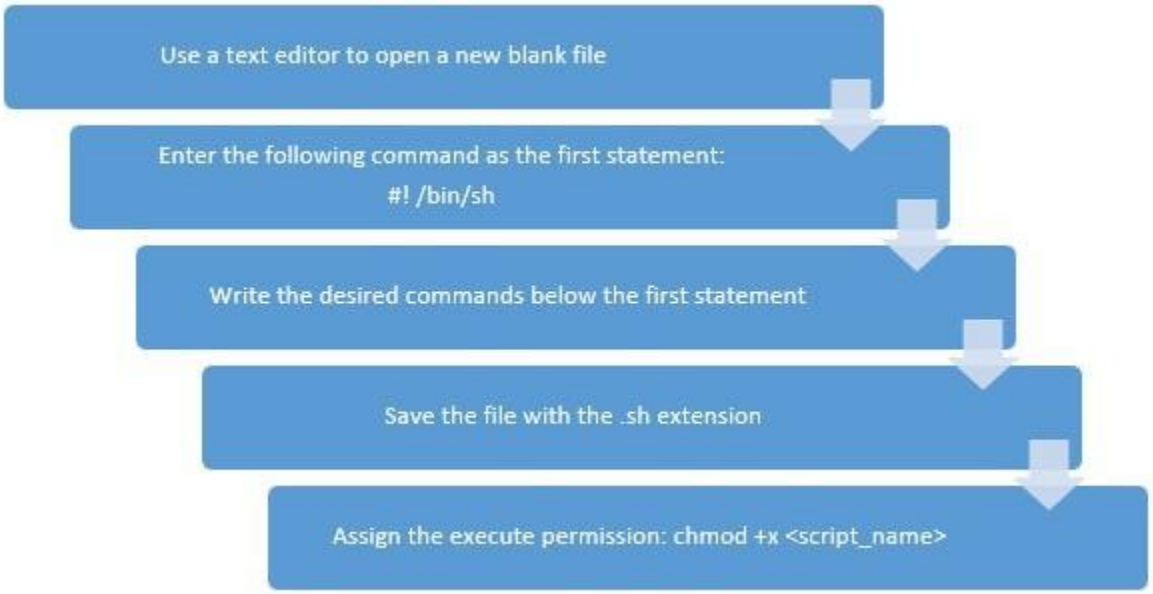
Figure 1.13 Screenshot of PLABLINUX01: Displaying the bashrc file.

Task 2 - Write a simple bash script

Assume that a user has to execute a series of commands multiple times. One method that the user can use is to keep typing in the commands over and over again. This method works well, but it is time-consuming. Another method that the user can use is to create a file that contains the series of command, which shell can execute without any manual intervention.

The file that the user will execute to automate the execution of multiple commands is known as a shell script. The user can create the shell script, store it, and execute it as many times as required. Using this method, the user does not have to re-type the same commands.

A user needs to perform the following steps to create and execute a shell script:



To write a simple bash script, perform the following steps:

Step 1

Ensure the terminal window is opened on PLABLINUX01.

Clear the screen by entering the following command:

```
clear
```

To create a shell script, type the following command:

```
vi script.sh
```

Press Enter.

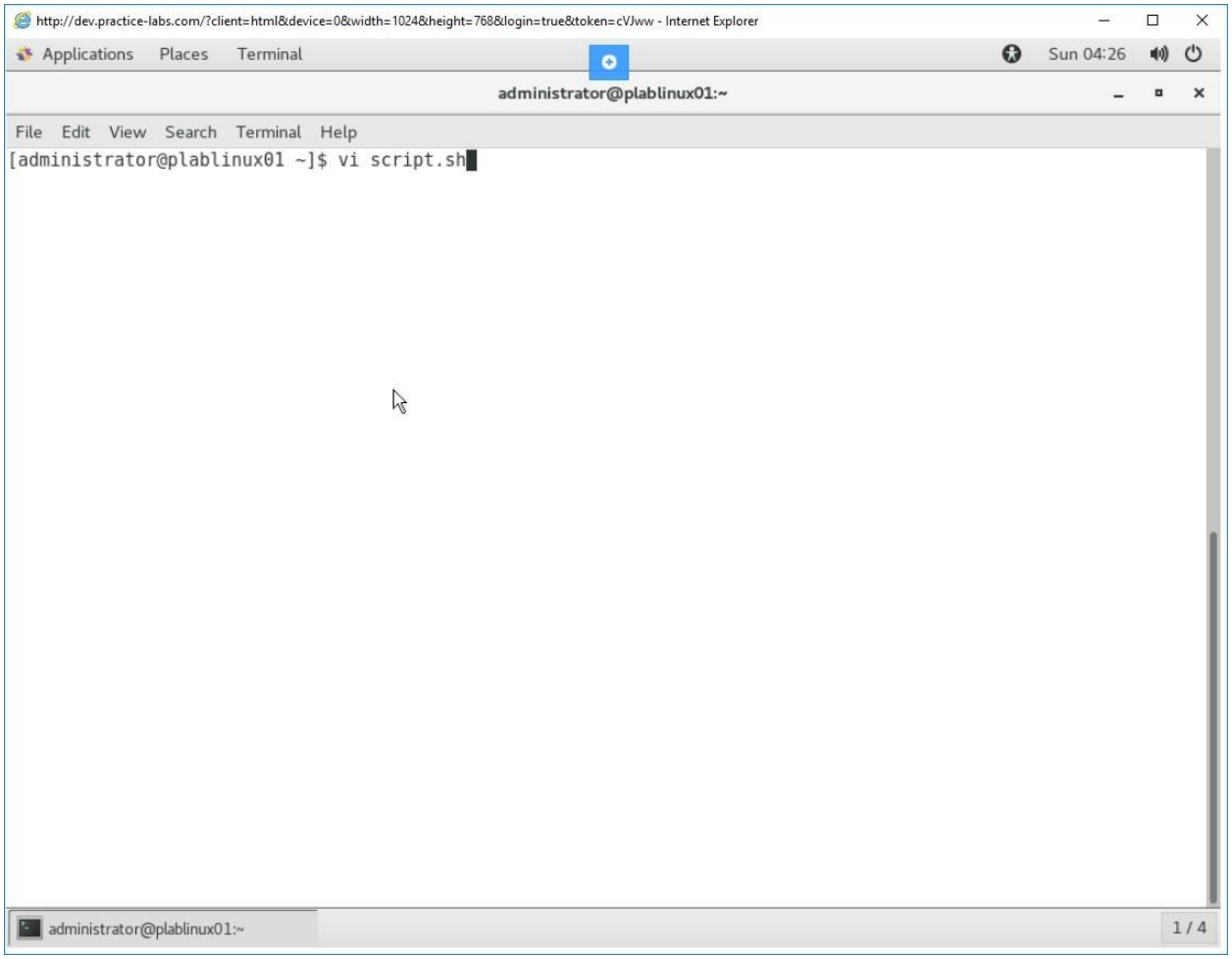


Figure 1.14 Screenshot of PLABLINUX01: Creating a new script using the vi editor.

Step 2

Press i to start the insert mode.

#!/bin/sh will always be the first line that tells the operating system that the script needs to be executed by the Bourne shell. In this case, it is mentioned as #!/bin/sh, which is the default location of the Bourne shell.

Type the following statement:

```
#!/bin/sh
```

Press Enter.

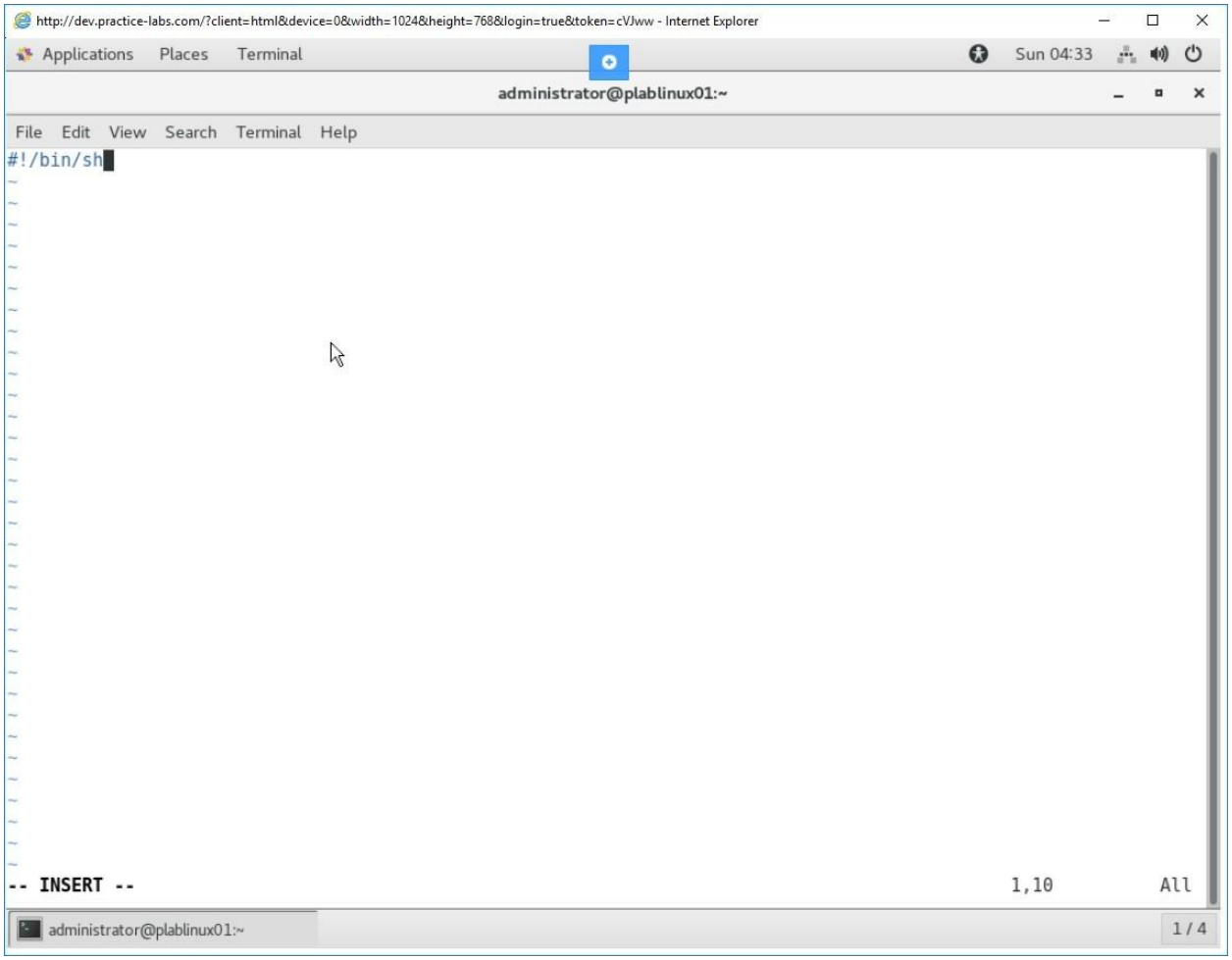


Figure 1.15 Screenshot of PLABLINUX01: Entering the statements in the shell script.

Step 3

Type the following statement:

```
echo "Good  Morning!"
```

Press Enter.

Note: After typing Good, press the tab key.

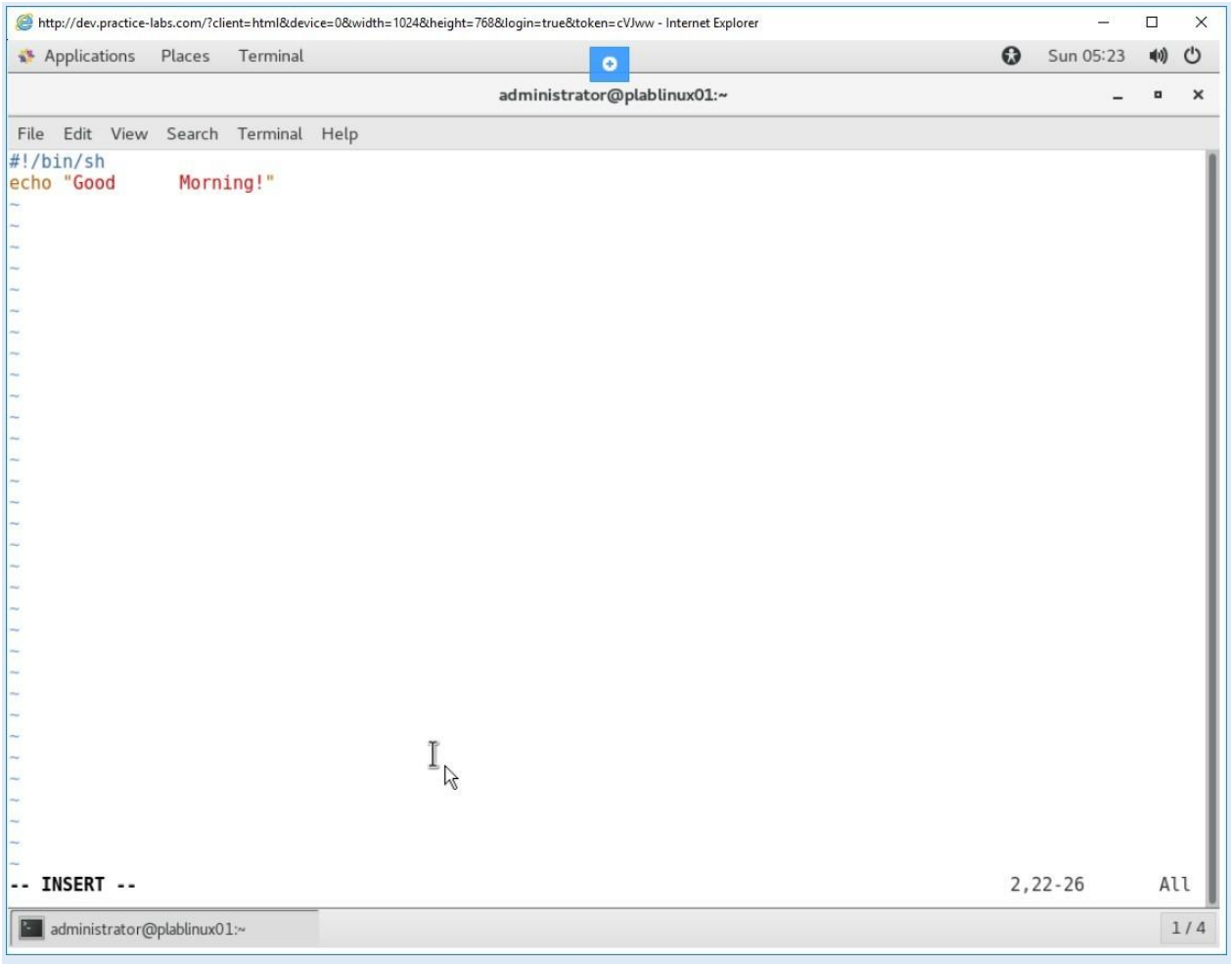


Figure 1.16 Screenshot of PLABLINUX01: Entering the statements in the shell script.

Step 4

To save the file, you need to exit the insert mode. Press ESC. Type the following statement:

```
:wq
```

Press Enter.

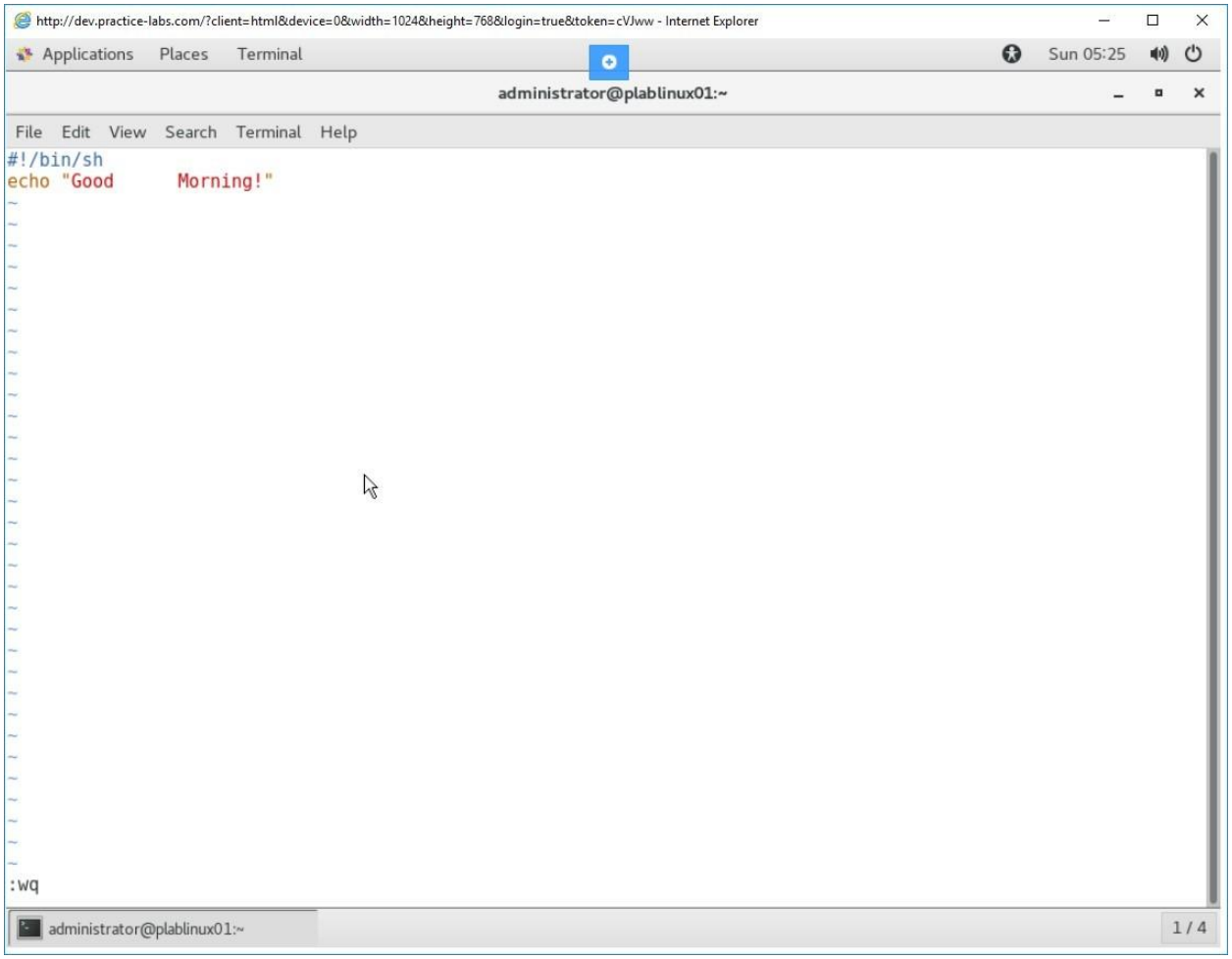


Figure 1.17 Screenshot of PLABLINUX01: Saving and closing the shell script.

Step 5

You are now in the terminal window.

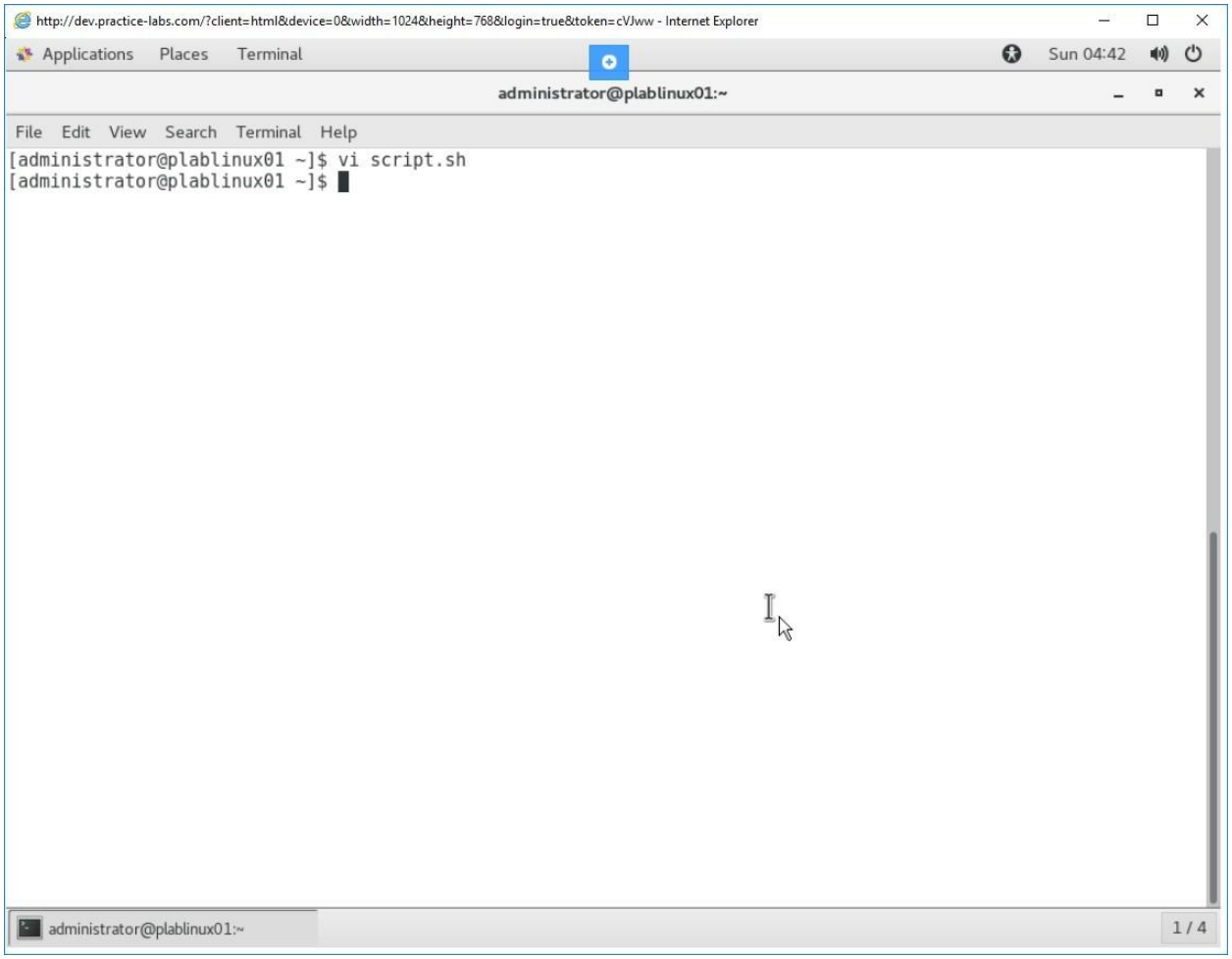


Figure 1.18 Screenshot of PLABLINUX01: Displaying the terminal window after shell script closure.

Step 6

Next, you need to change the permissions on this shell script. Before a shell script can be executed, the user must make the shell script executable. To do this, type the following command:

```
chmod 755 script.sh
```

Press Enter.

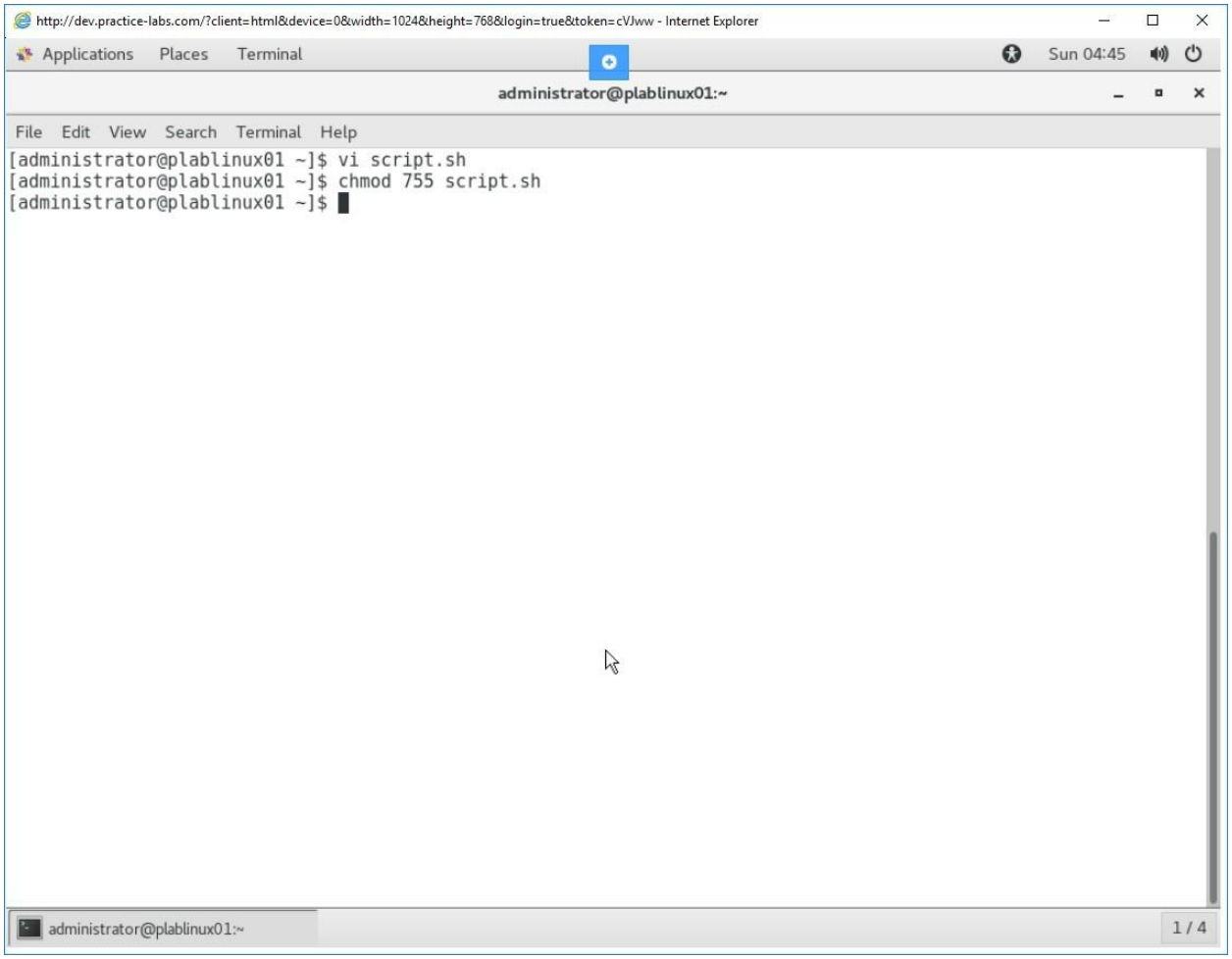


Figure 1.19 Screenshot of PLABLINUX01: Assigning execute permissions to the script.

Step 7

After the script is made executable, it can be executed in two different ways. The first method is to use the bash command. To execute the script, type the following command:

```
bash script.sh
```

Press Enter.

Note: With the use of echo command, you need to put the string of characters in the quote. It will be printed in the verbatim manner as you have entered. Remember that you had pressed tab after Good.

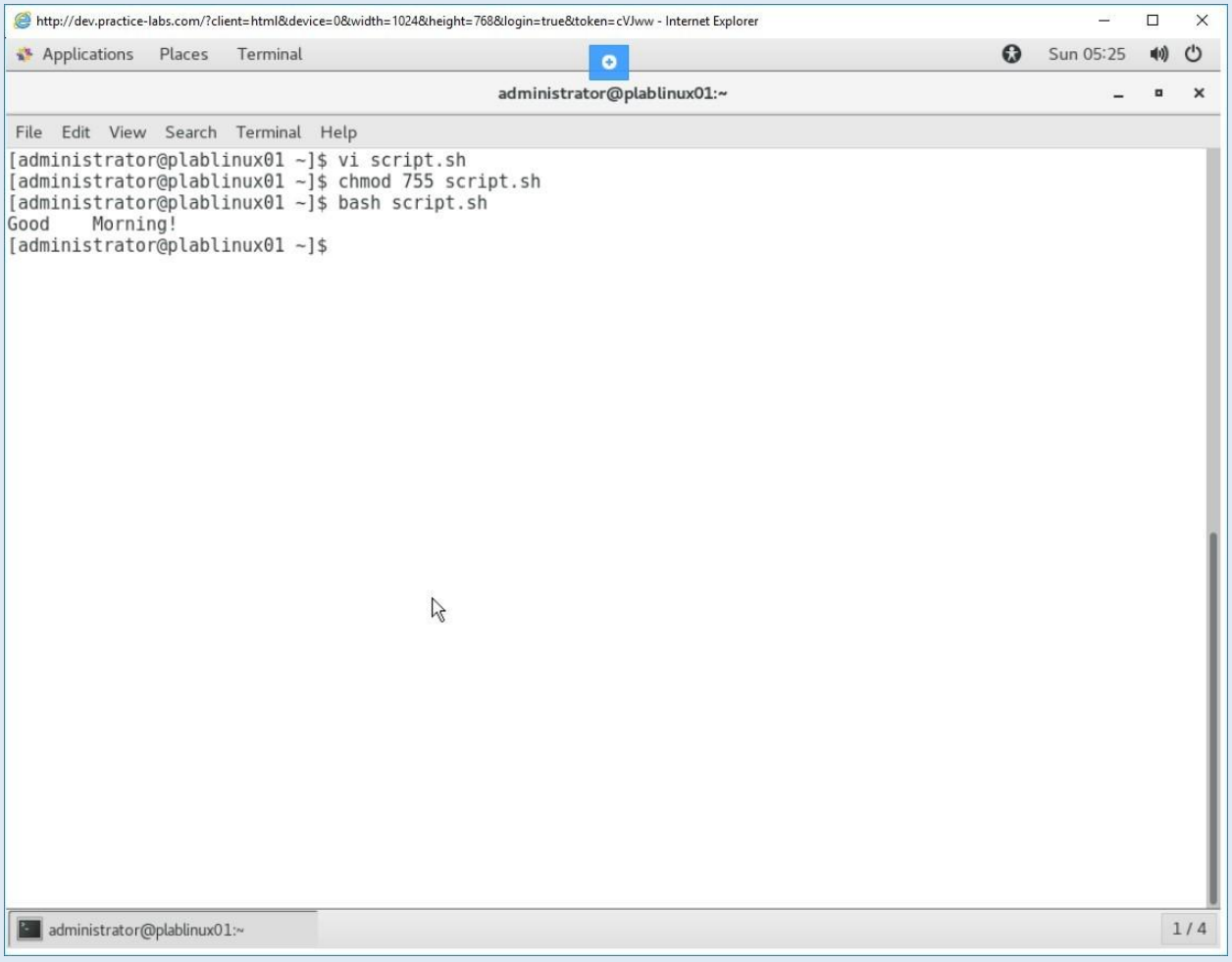


Figure 1.20 Screenshot of PLABLINUX01: Executing the script.

Step 8

The second method to execute a script is to prefix it with ./. Type the following command:

```
./script.sh
```

Press Enter.

Notice that the output of both the method is just the same.

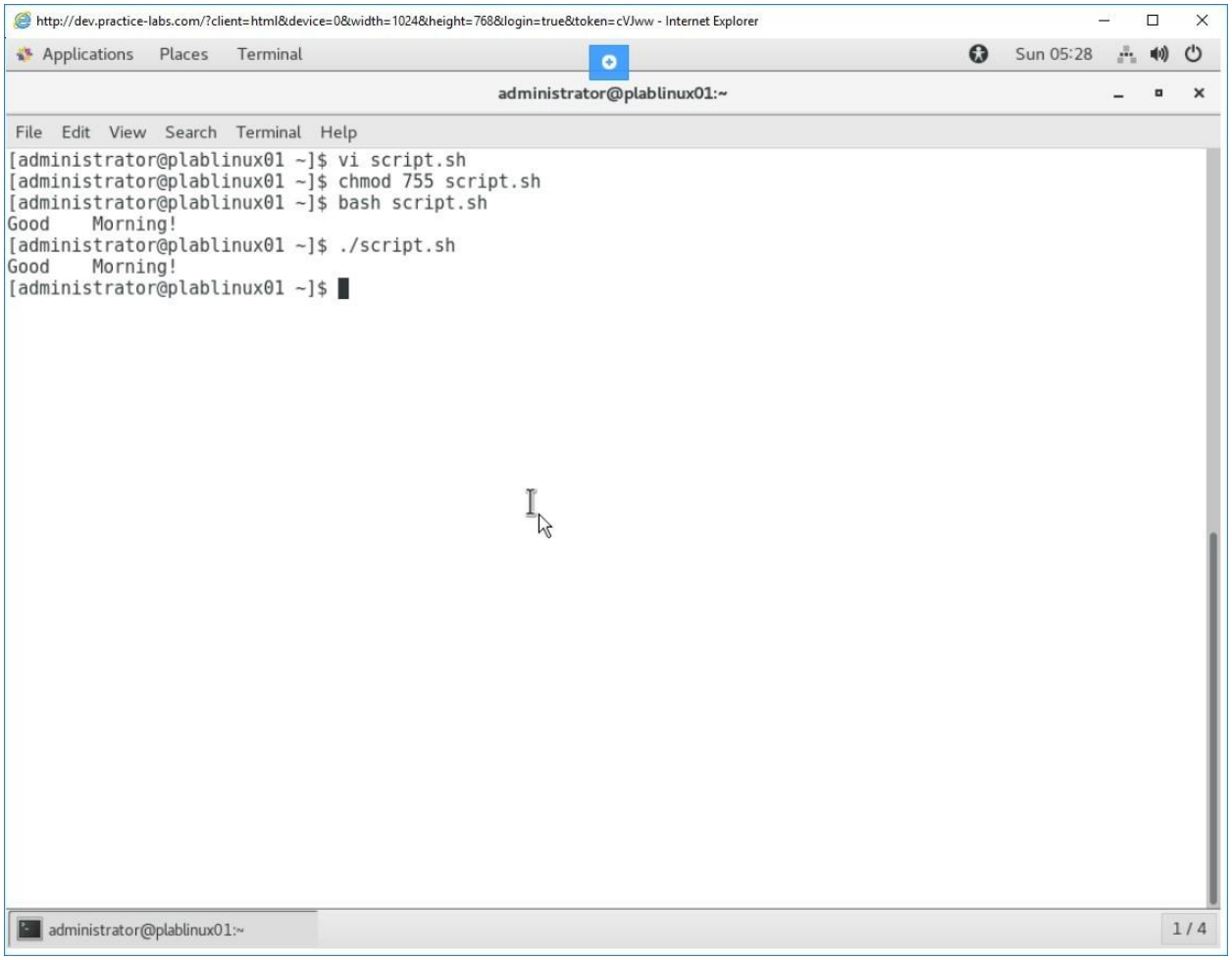


Figure 1.21 Screenshot of PLABLINUX01: Executing the script.

Step 9

Clear the screen by entering the following command:

```
clear
```

You can also use a third method that is recommended by various Linux flavors. You should create a subdirectory named bin in your home directory. Then, move the script to the subdirectory. To create the subdirectory, type the following command:

```
mkdir bin
```

Press Enter.

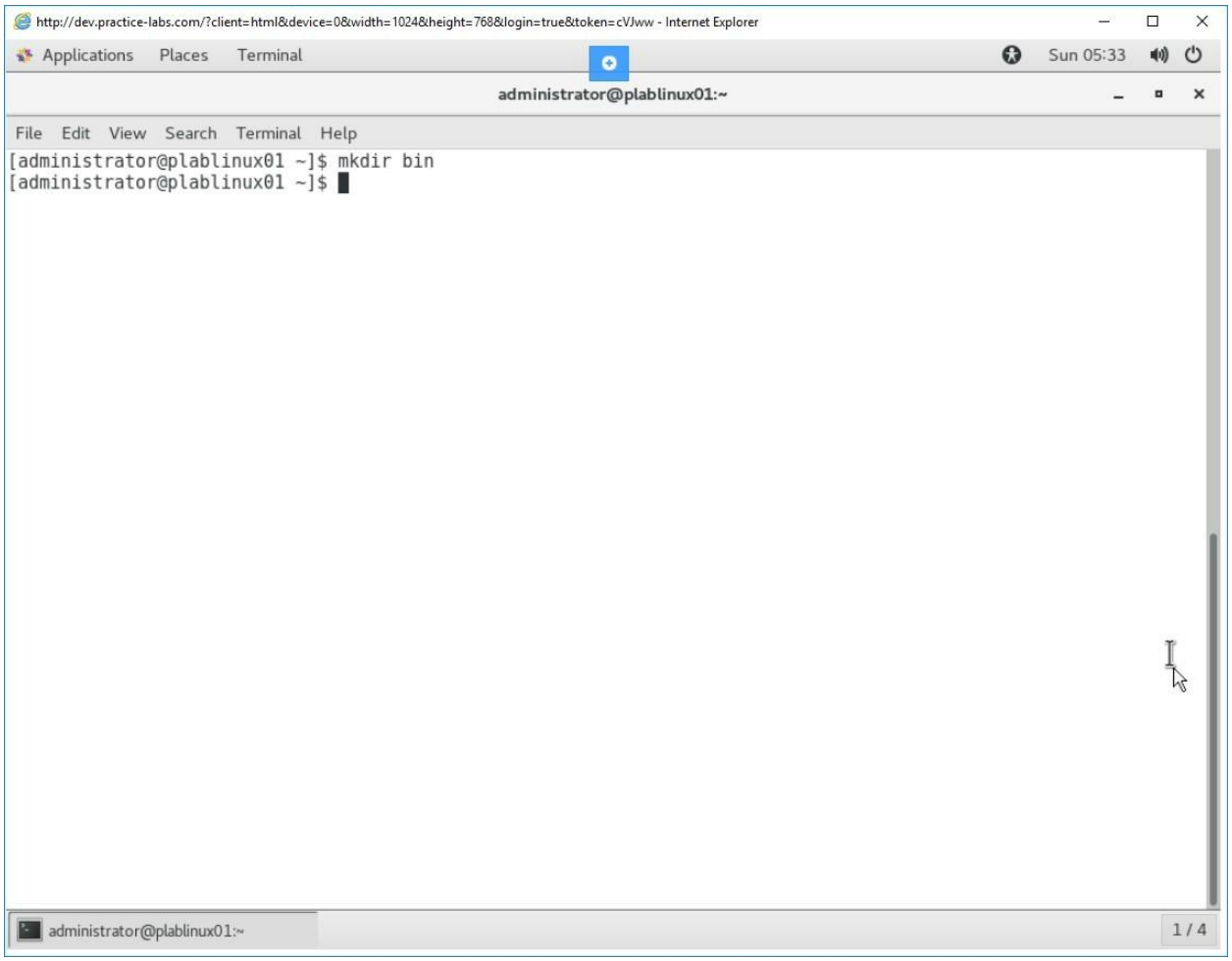


Figure 1.22 Screenshot of PLABLINUX01: Creating a new directory.

Step 10

Clear the screen by entering the following command:

```
clear
```

To list the files in the home directory, type the following command:

```
ls
```

Press Enter.

Move the file to the bin directory. Type the following command:

```
mv script.sh bin
```

Press Enter.

To list the files in the home directory, type the following command:

```
ls
```

Press Enter.

Notice that script.sh has moved to the bin folder.

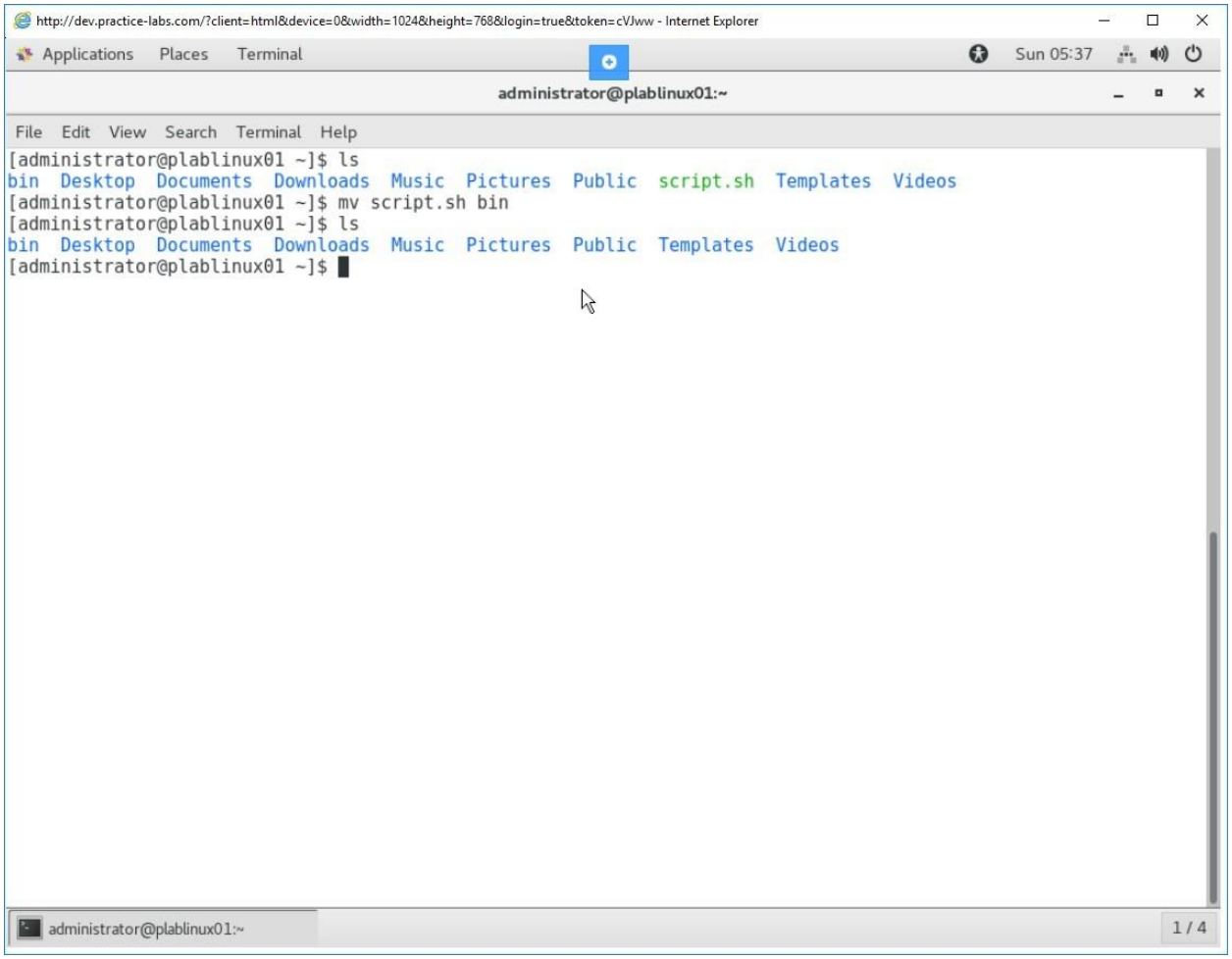


Figure 1.23 Screenshot of PLABLINUX01: Moving the file into the bin directory.

Step 11

Clear the screen by entering the following command:

```
clear
```

To navigate to the bin directory, type the following command:

```
cd bin
```

Press Enter.

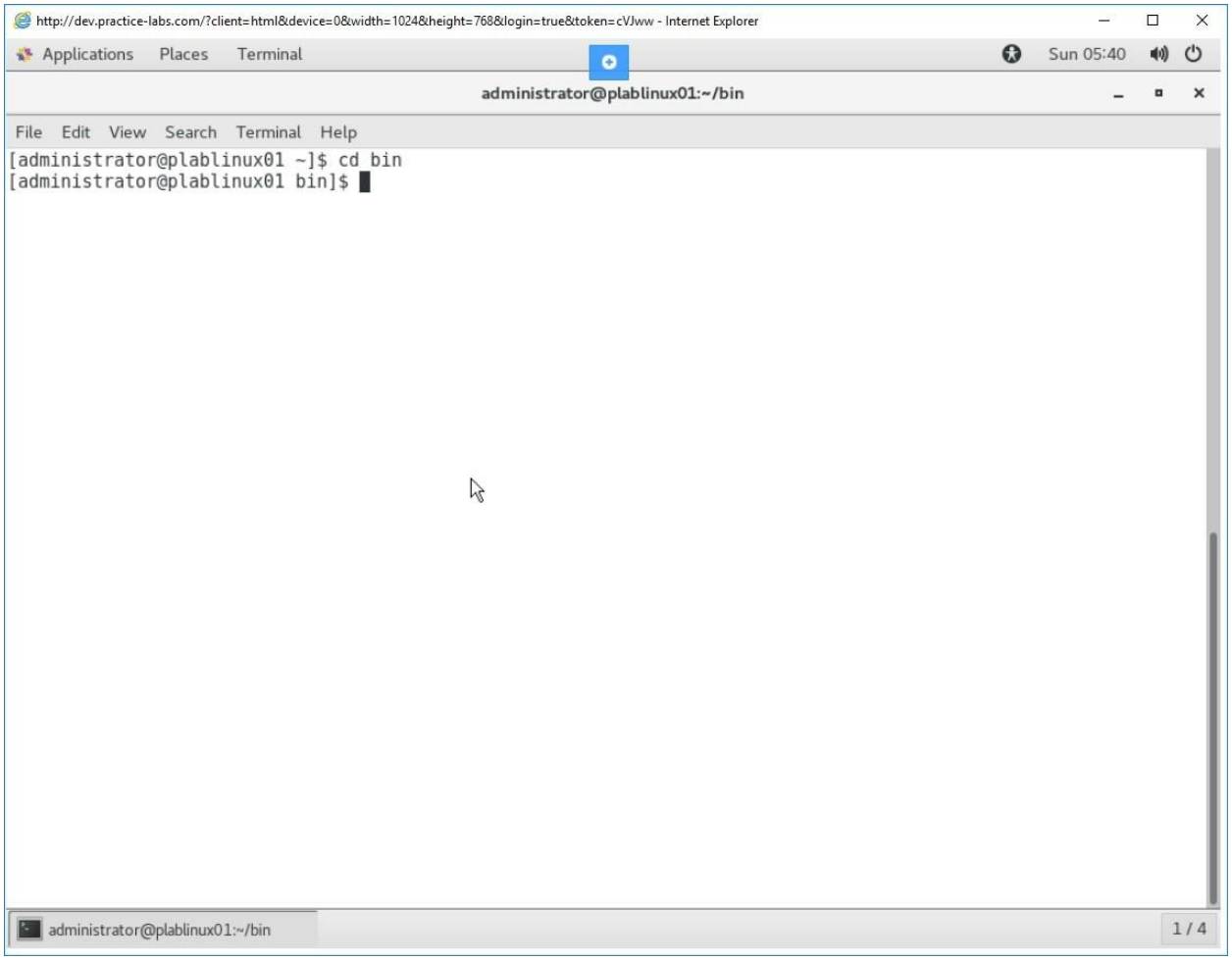


Figure 1.24 Screenshot of PLABLINUX01: Changing to the bin directory.

Step 12

To execute the script, type the following command:

```
script.sh
```

Press Enter.

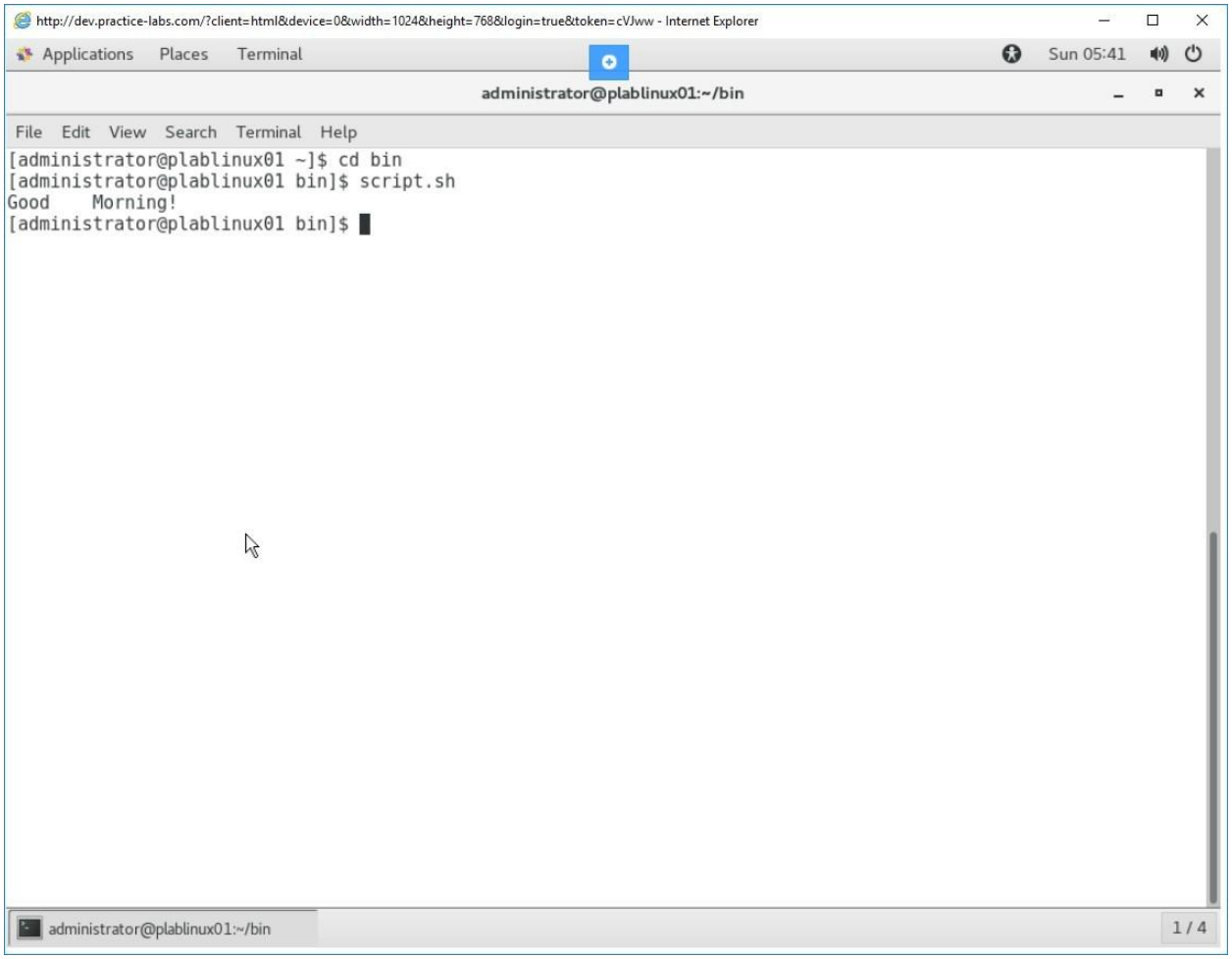


Figure 1.25 Screenshot of PLABLINUX01: Executing the script.

Task 3 - Use Commenting

Scripts can also have comments, which are statements that do not get executed.

To add comments in the script.sh file, perform the following steps:

Step 1

Clear the screen by entering the following command:

```
clear
```

Open the script.sh file once again. Type the following command:

```
vi script.sh
```

Press Enter.

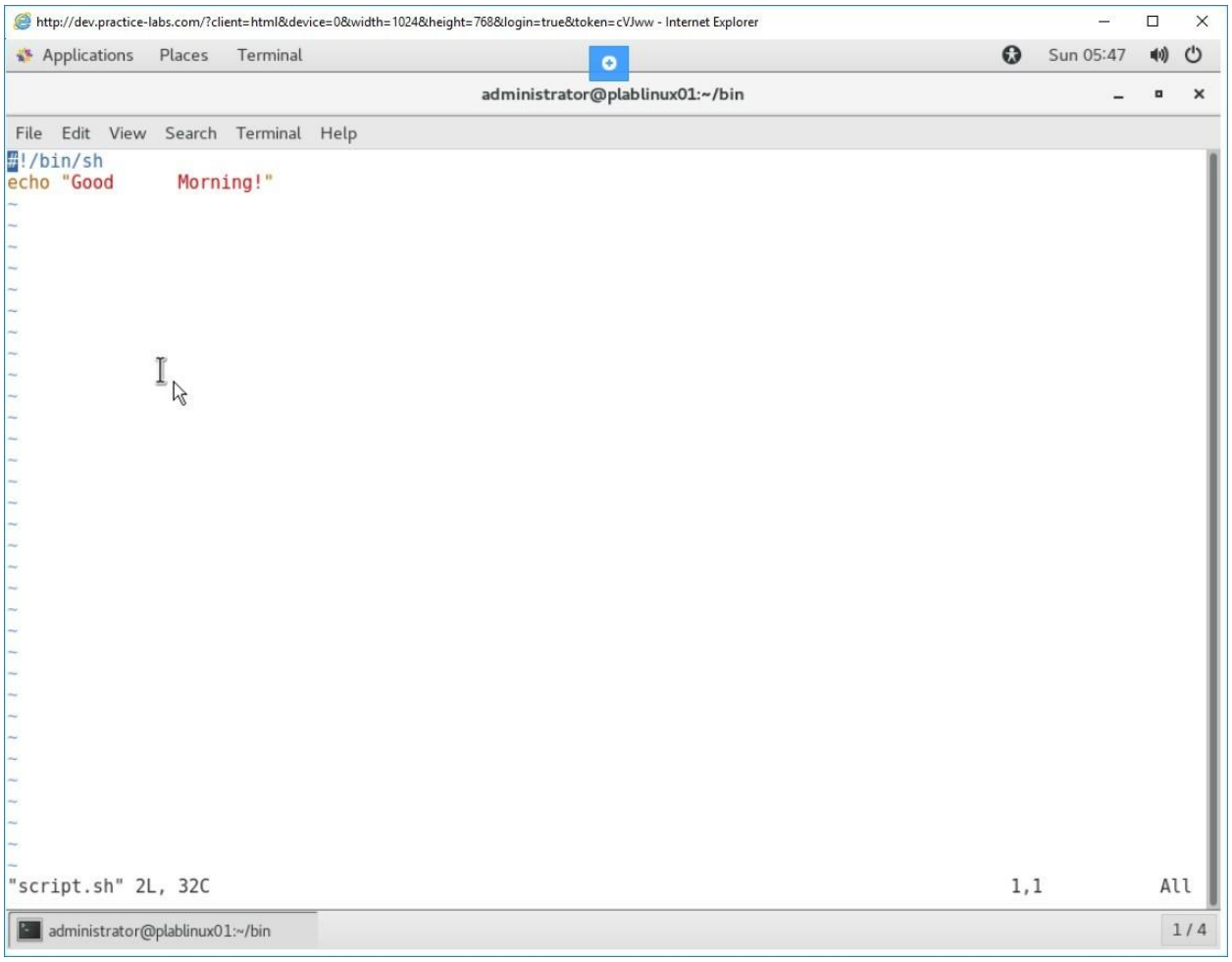


Figure 1.26 Screenshot of PLABLINUX01: Opening the script in the vi editor.

Step 2

Press i to start the insert mode. Scroll down to the next blank line.

Type the following statement:

```
#This is just a test script.
```

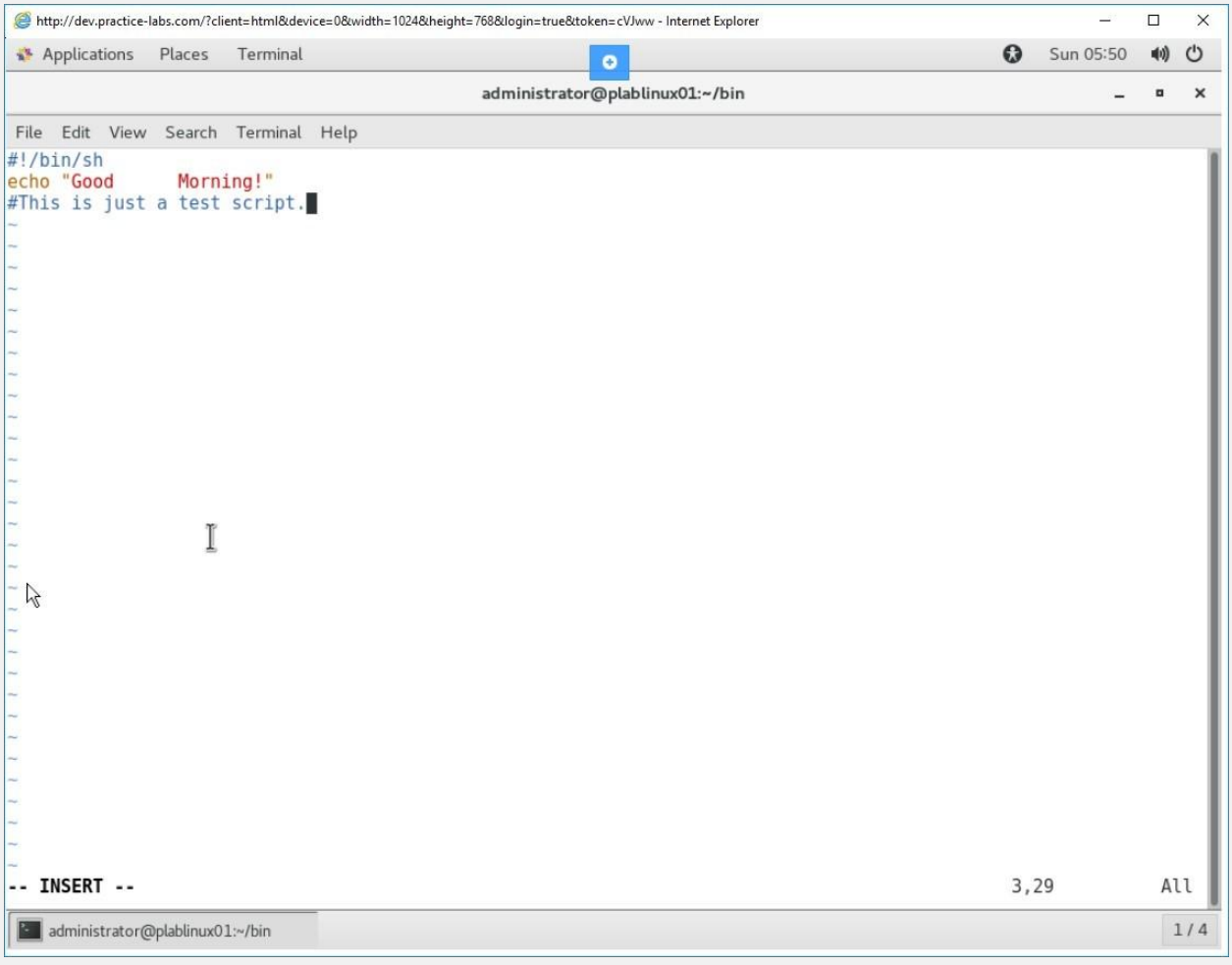


Figure 1.27 Screenshot of PLABLINUX01: Adding a comment statement in the script.

Step 3

To save the file, you need to exit the insert mode. Press ESC. Type the following statement:

```
:wq
```

Press Enter.

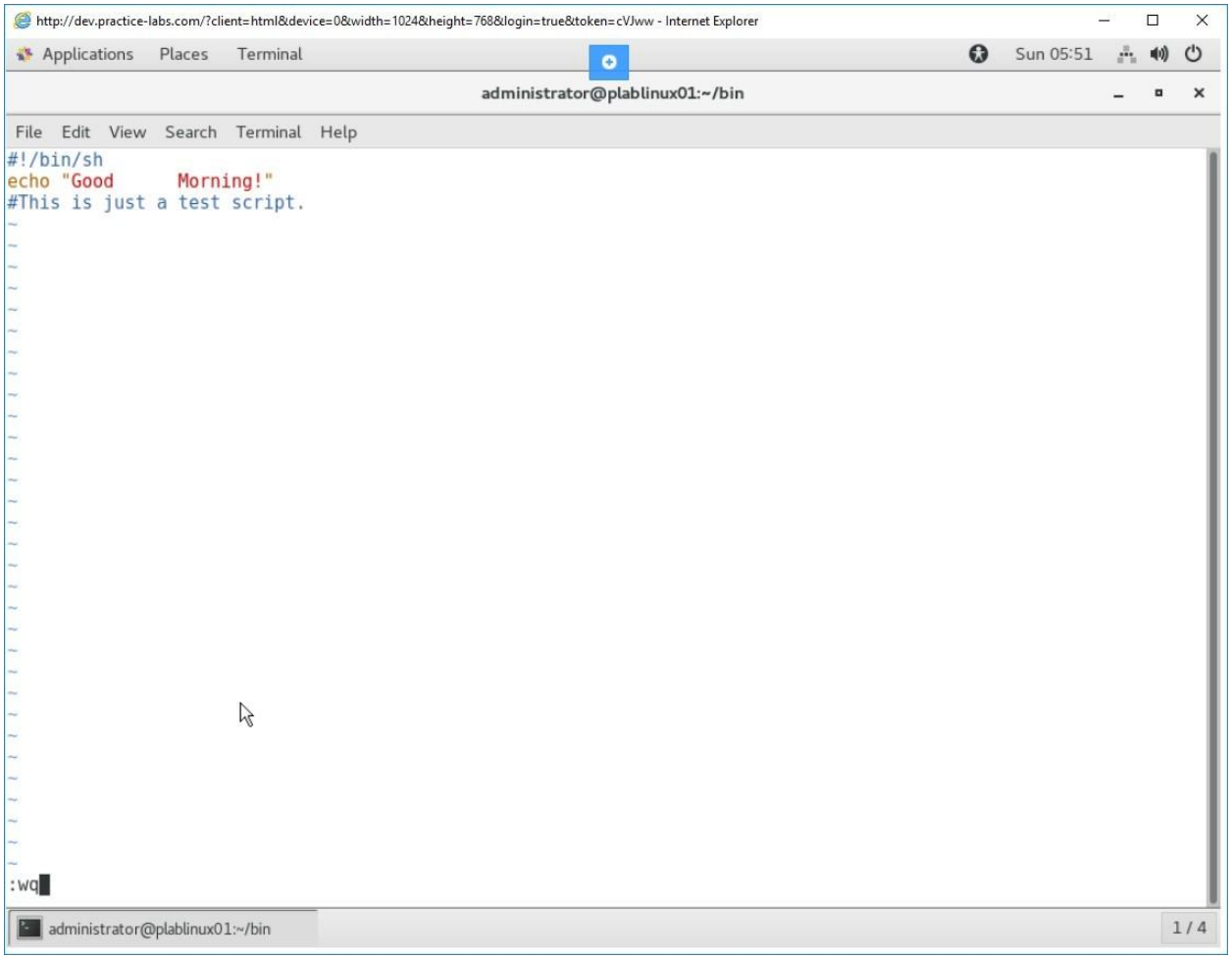


Figure 1.28 Screenshot of PLABLINUX01: Saving and closing the script.

Step 4

You are now in the terminal window.

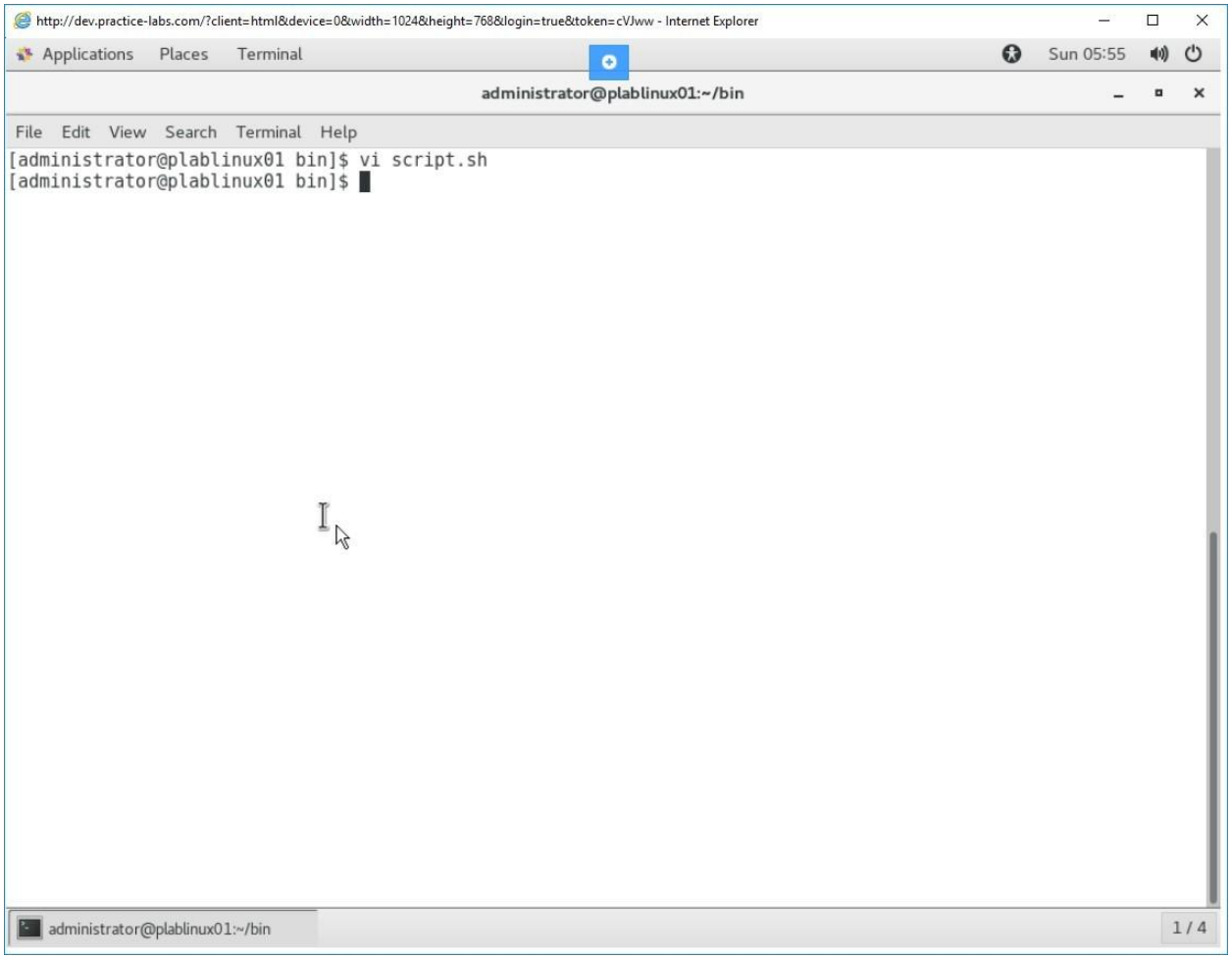


Figure 1.29 Screenshot of PLABLINUX01: Navigating back to the terminal window.

Step 5

Execute the script. Type the following command:

```
script.sh
```

Press Enter.

Notice that the output does not show any comment.

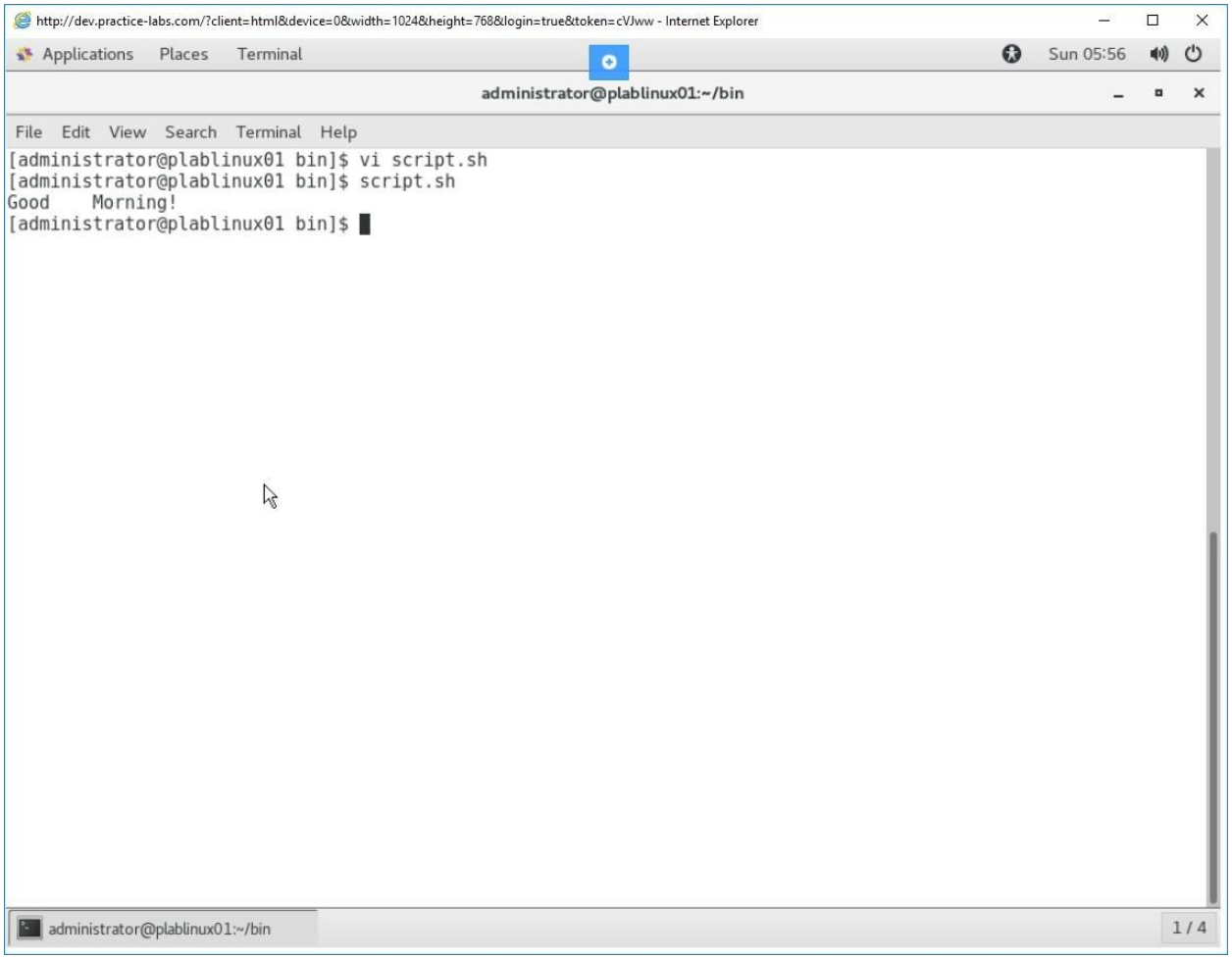


Figure 1.30 Screenshot of PLABLINUX01: Executing the script.

Task 4 - Use Parameters

Shell scripts, as stated earlier, can be written once and then re-used as many times as possible. The advantage of a shell script is also that you can pass different parameters to get different results.

To use parameters, perform the following steps:

Step 1

Clear the screen by entering the following command:

```
clear
```

Create the param.sh file. Type the following command:

```
vi param.sh
```

Press Enter.

Start the insert mode by pressing i.

Type the following in the script:

```
#!/bin/bash
echo "$# parameters"
echo "$@";
```

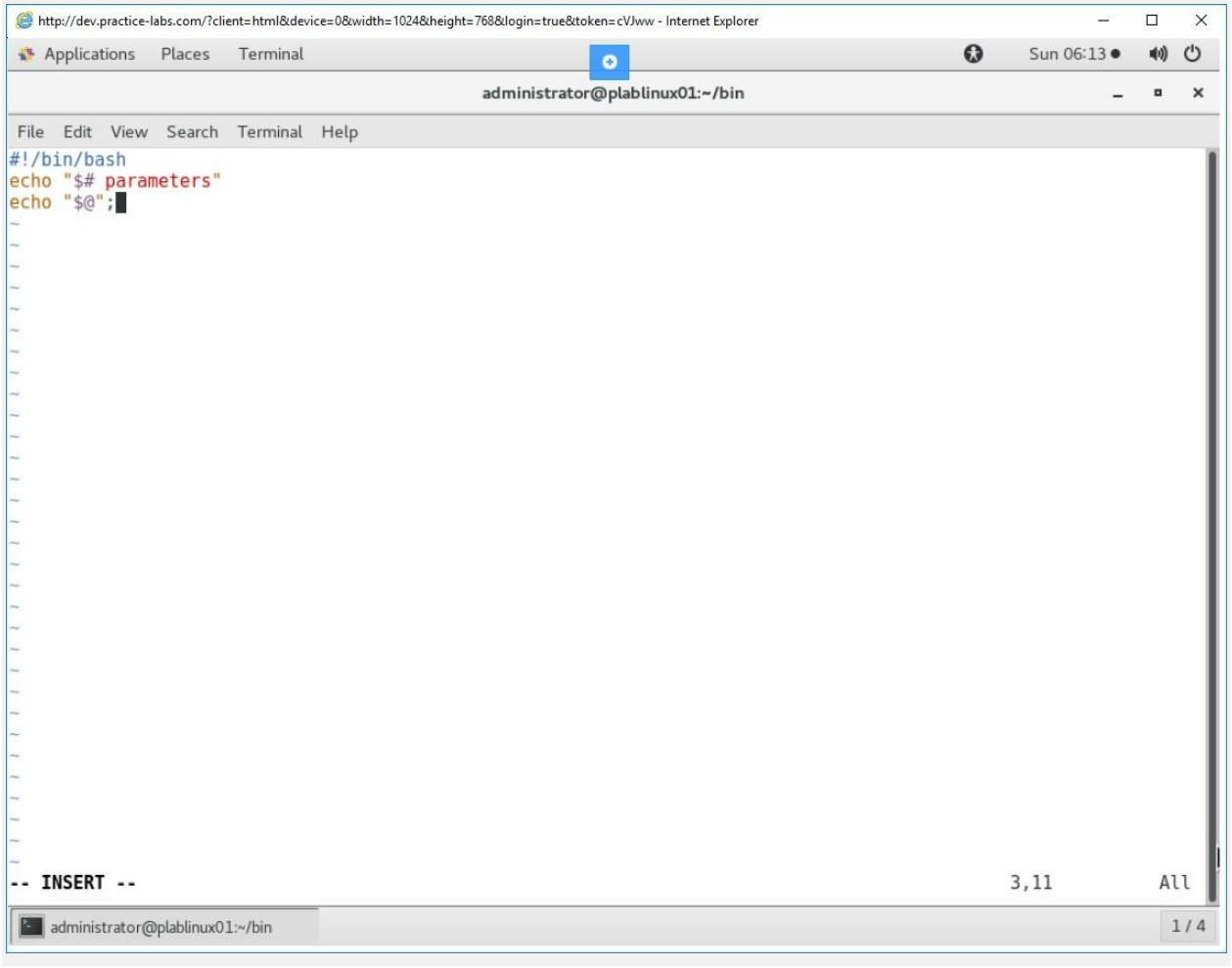


Figure 1.31 Screenshot of PLABLINUX01: Creating a new shell script using the vi editor.

Step 2

Press ESC and then type the following command:

```
:wq
```

Press Enter.

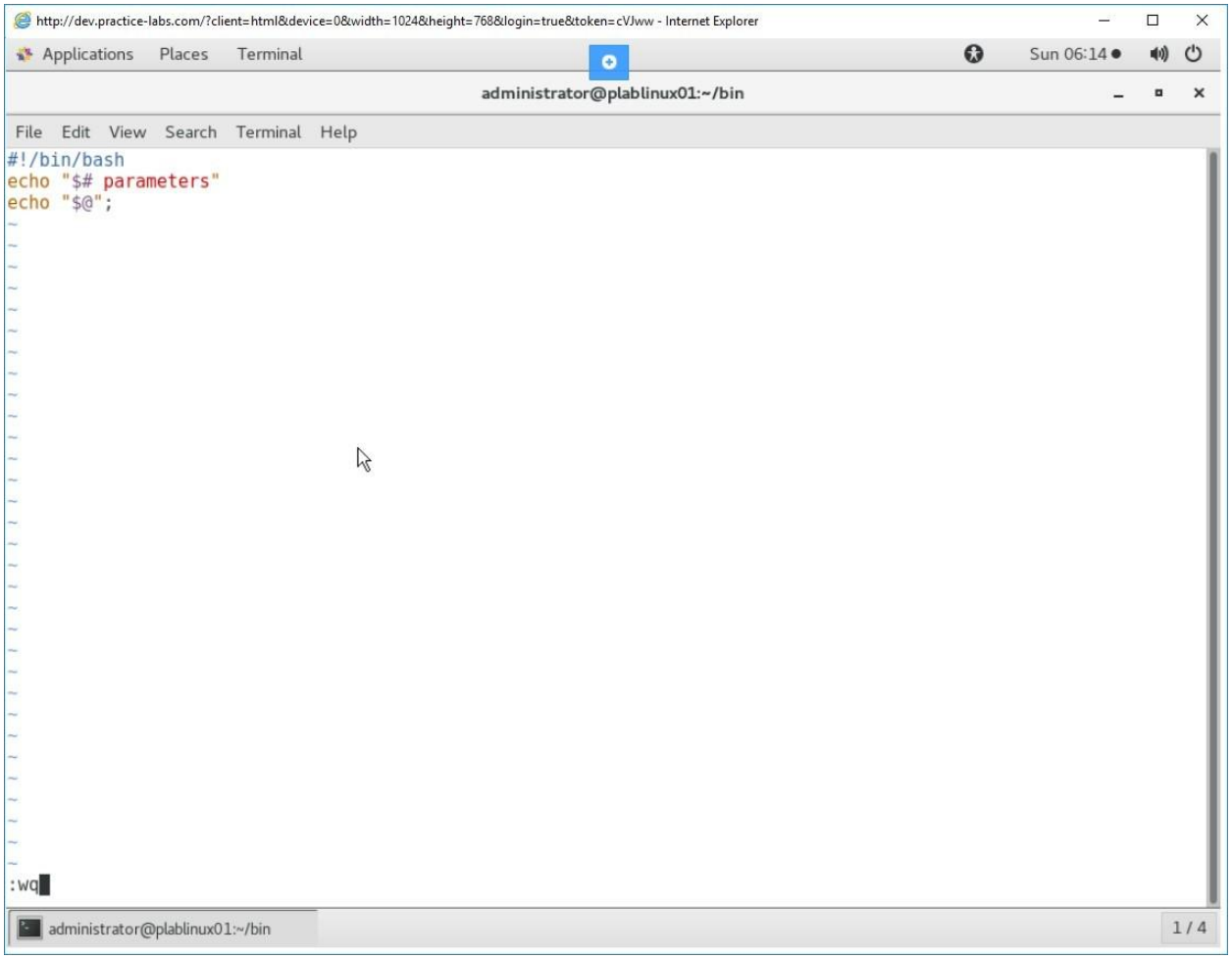


Figure 1.32 Screenshot of PLABLINUX01: Saving and closing the script.

Step 3

You are now back on the command prompt.

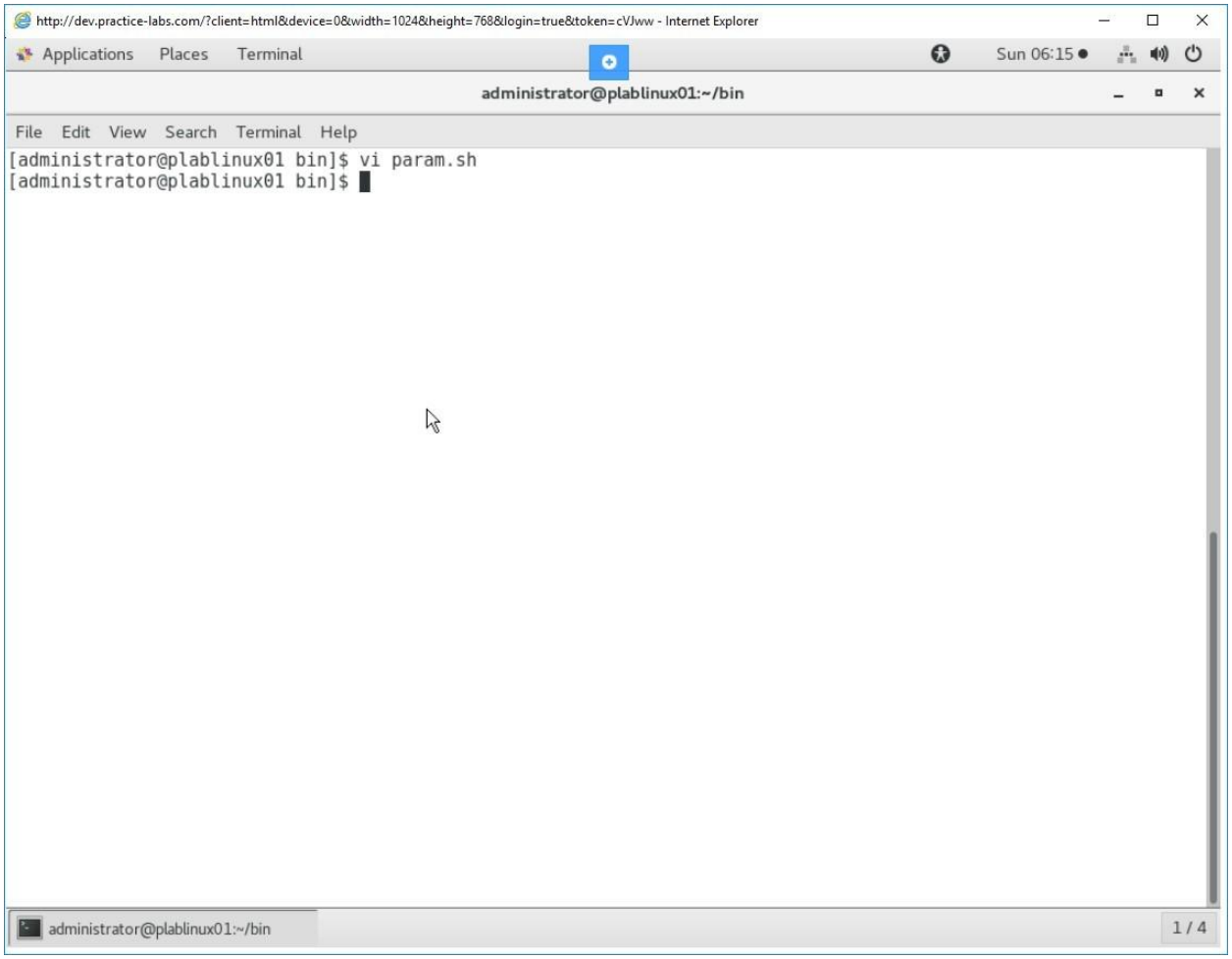


Figure 1.33 Screenshot of PLABLINUX01: Navigating back to the terminal window.

Step 4

Next, you need to change the permissions on this shell script. Before a shell script can be executed, the user must make the shell script executable. To do this, type the following command:

```
chmod 755 param.sh
```

Press Enter.

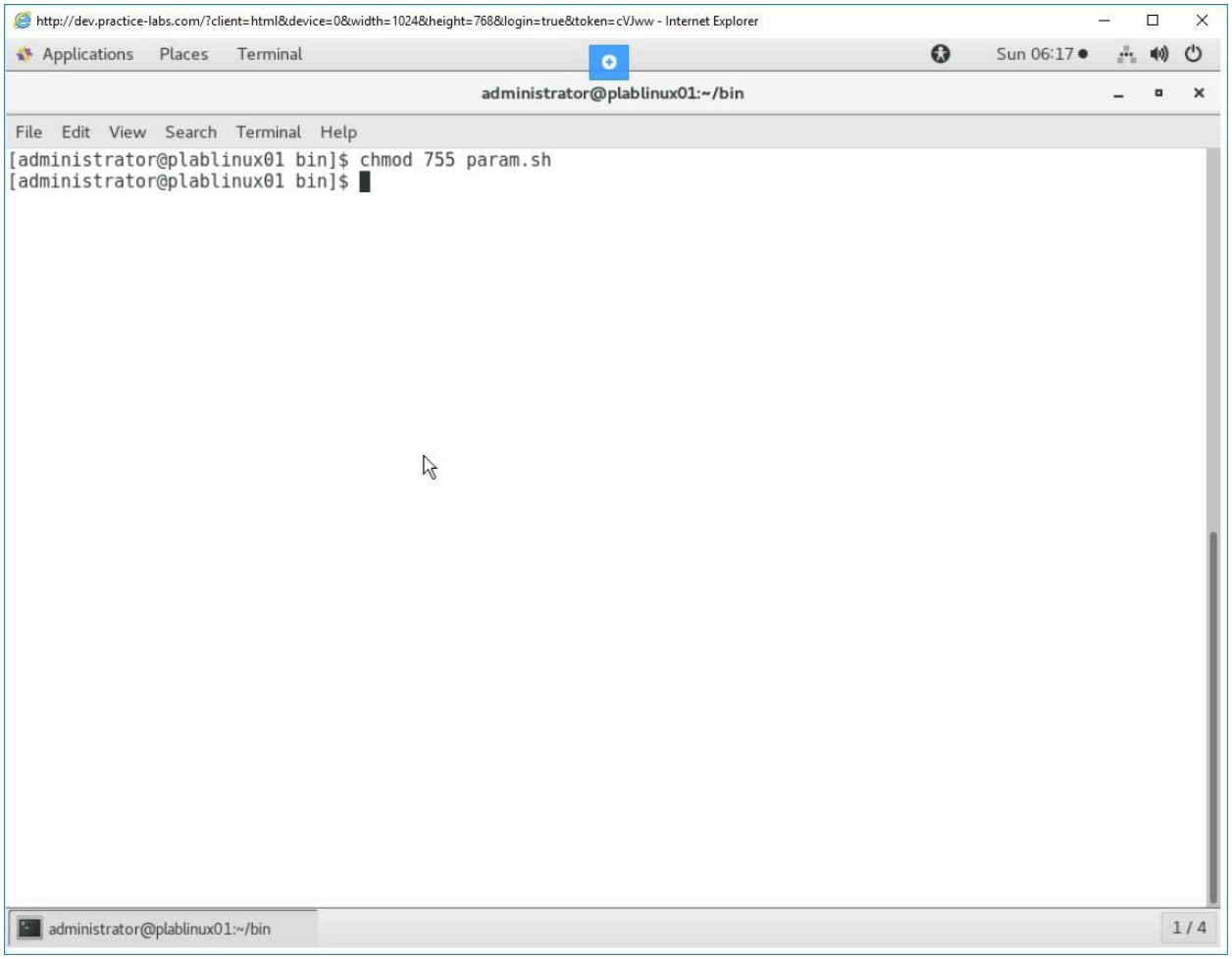


Figure 1.34 Screenshot of PLABLINUX01: Assigning the execute permission.

Step 5

You will now see how the script provides different results when parameters are passed on.

Type the following command:

```
param.sh 1 2 3
```

Press Enter.

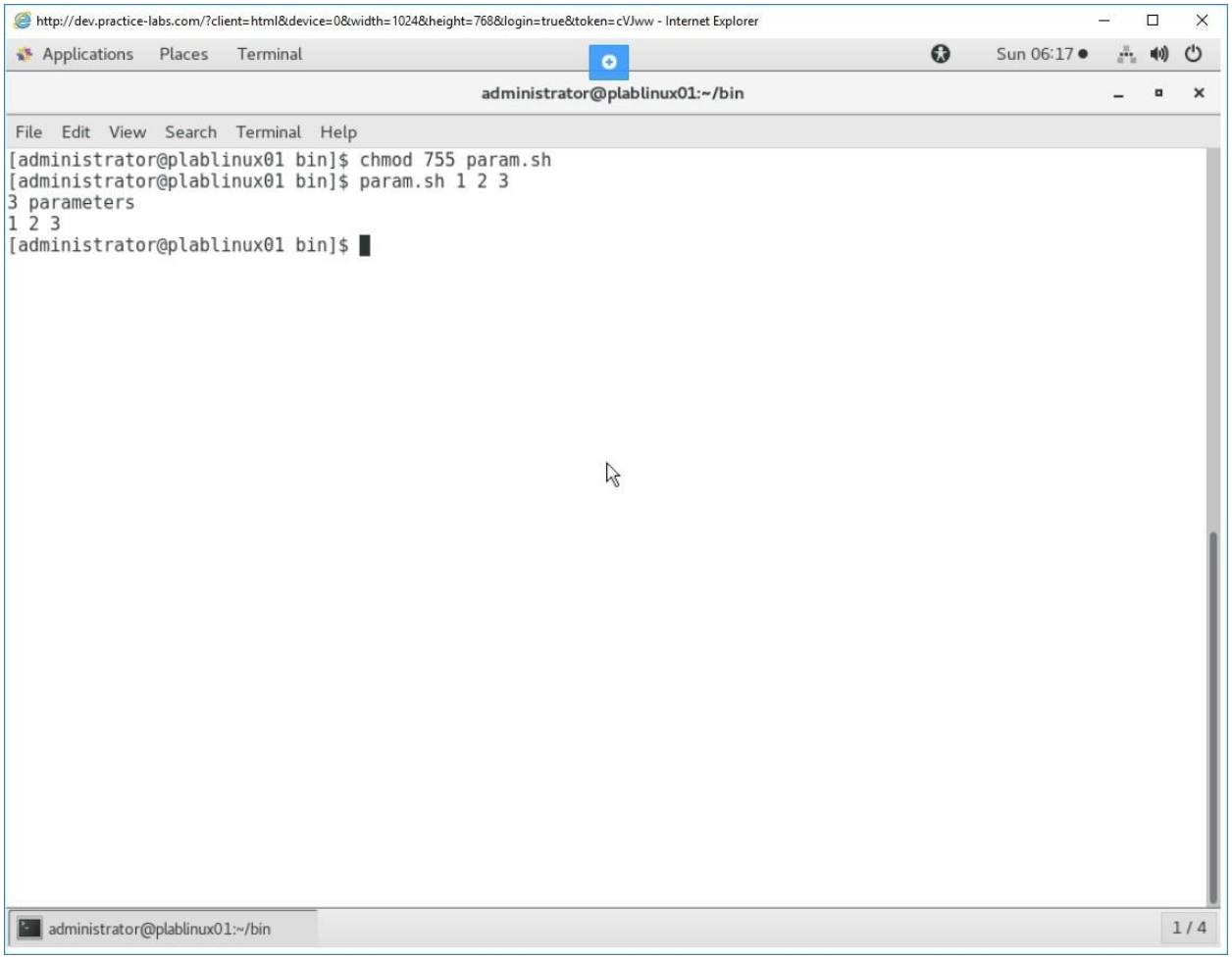


Figure 1.35 Screenshot of PLABLINUX01: Executing the shell script with the parameters.

Step 6

Type the following command:

```
param.sh "1 2 3"
```

Press Enter.

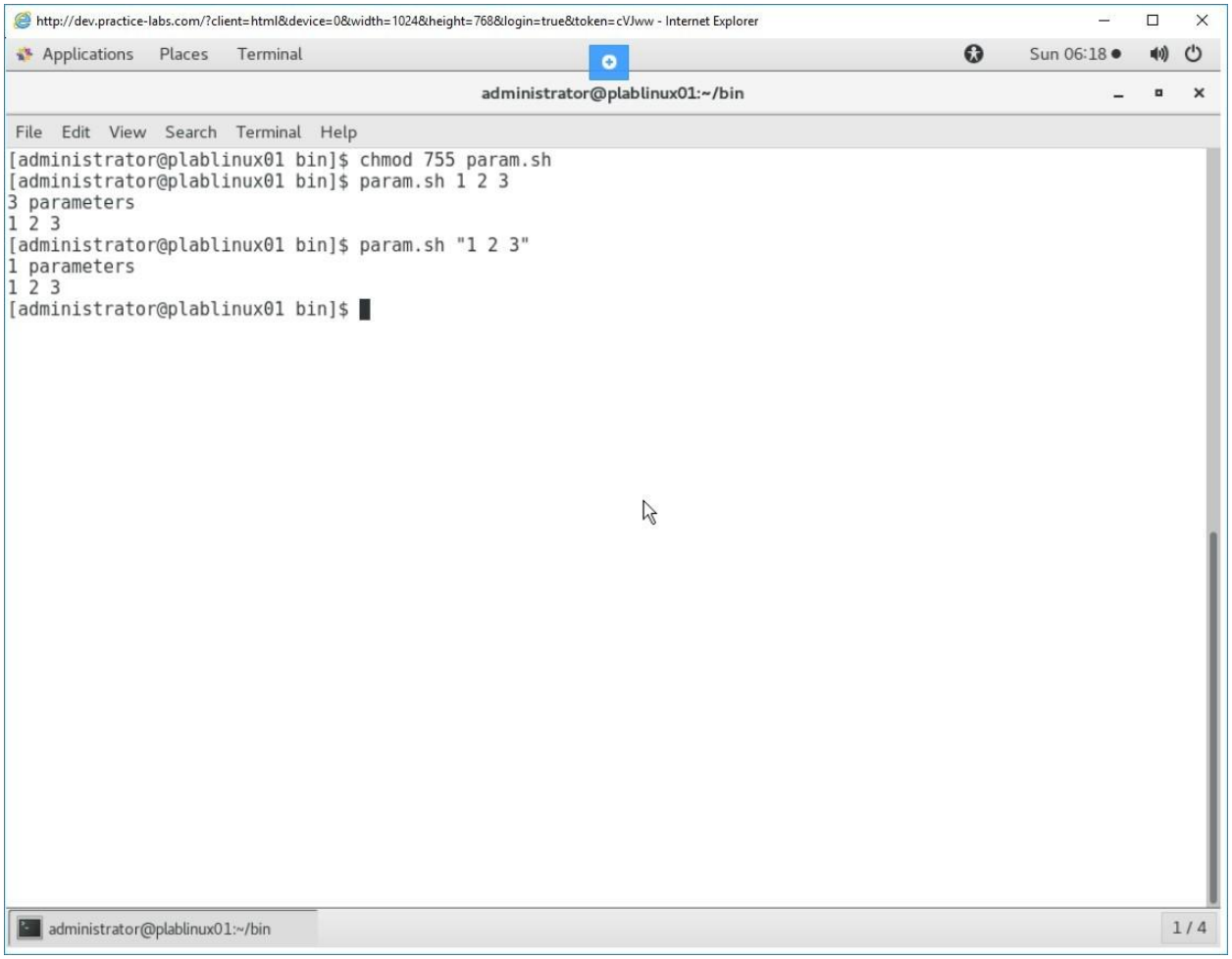


Figure 1.36 Screenshot of PLABLINUX01: Executing the shell script with the parameters.

Step 7

Type the following command:

```
param.sh 1 "2 3"
```

Press Enter.

Notice that in all three cases, the output was different.

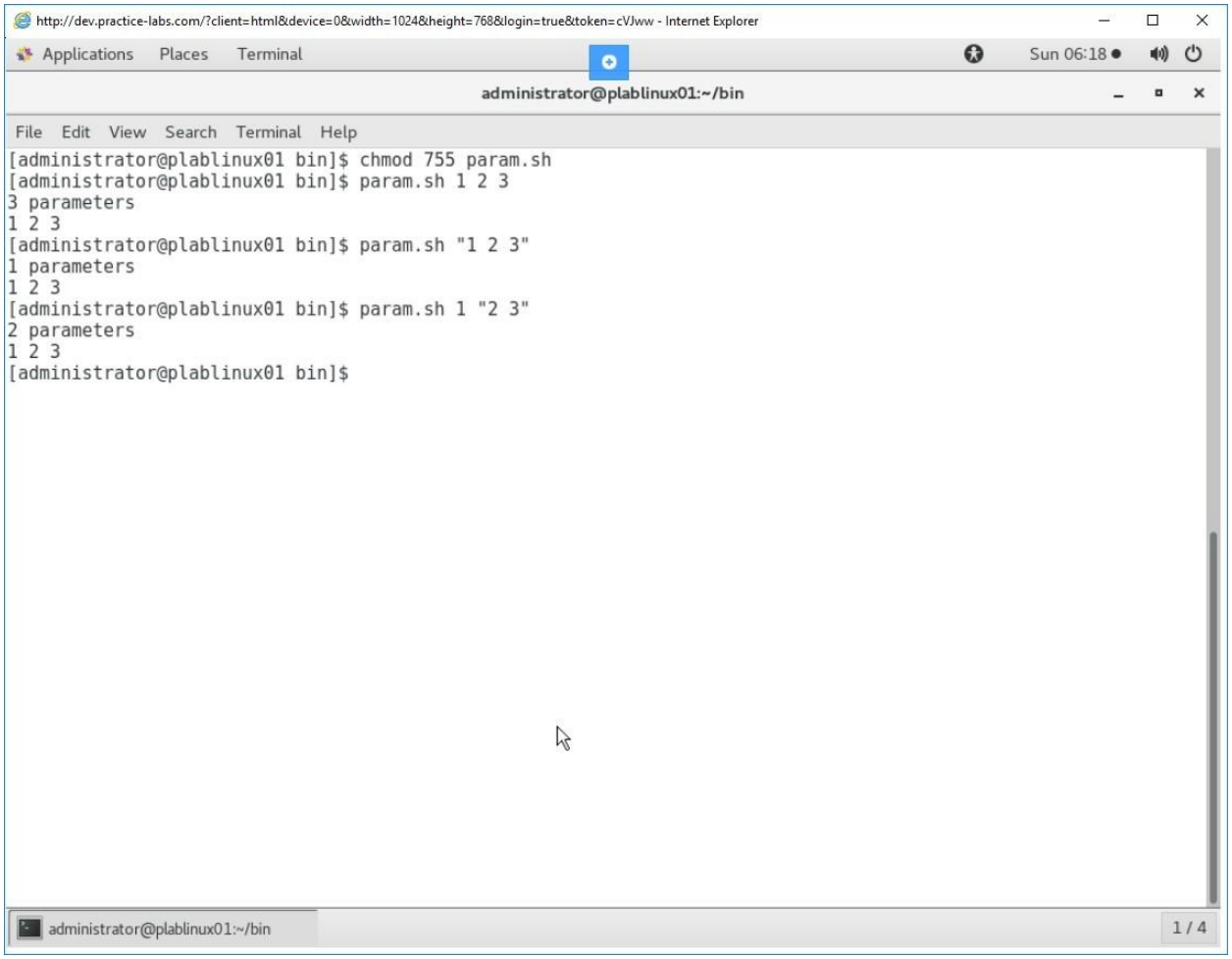


Figure 1.37 Screenshot of PLABLINUX01: Executing the shell script with the parameters.

Task 5 - Capture User Inputs in Scripts

You can also capture user inputs in scripts. For example, you can prompt a user to enter the name and then display the name.

To capture user inputs in scripts, perform the following steps:

Step 1

Create another script named test.sh. Type the following command:

Type the following in the script using the insert mode:

```
#!/bin/bash
echo Hello, Please enter your name.
read varname
echo Hello $varname
```

Note: The value entered by the user is stored in the variable named varname.

Press ESC and then type the following command:

```
:wq
```

Press Enter.

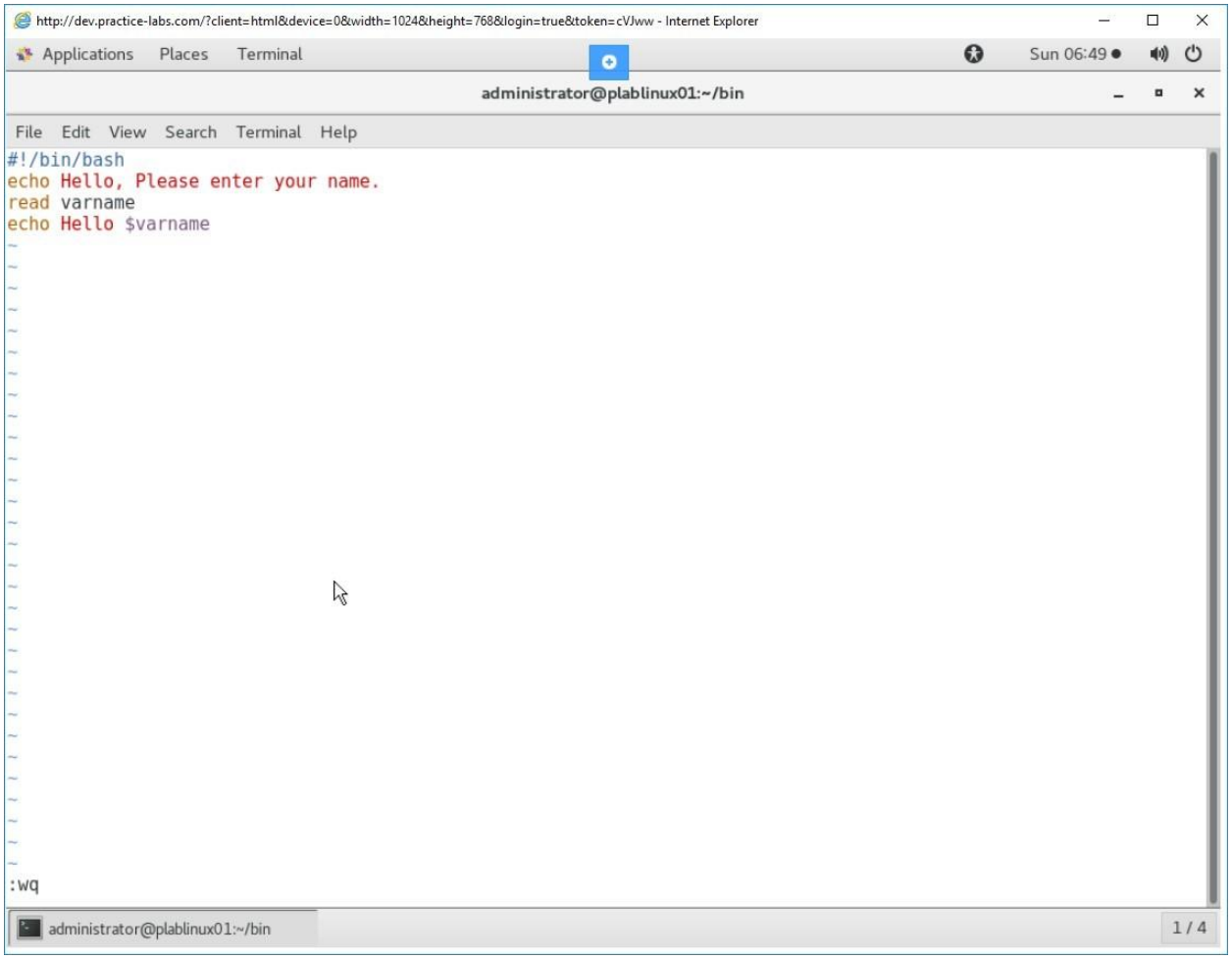


Figure 1.38 Screenshot of PLABLINUX01: Creating a new shell script using the vi editor.

Step 2

Back on the terminal window, type the following command:

```
chmod 755 test.sh
```

Press Enter.

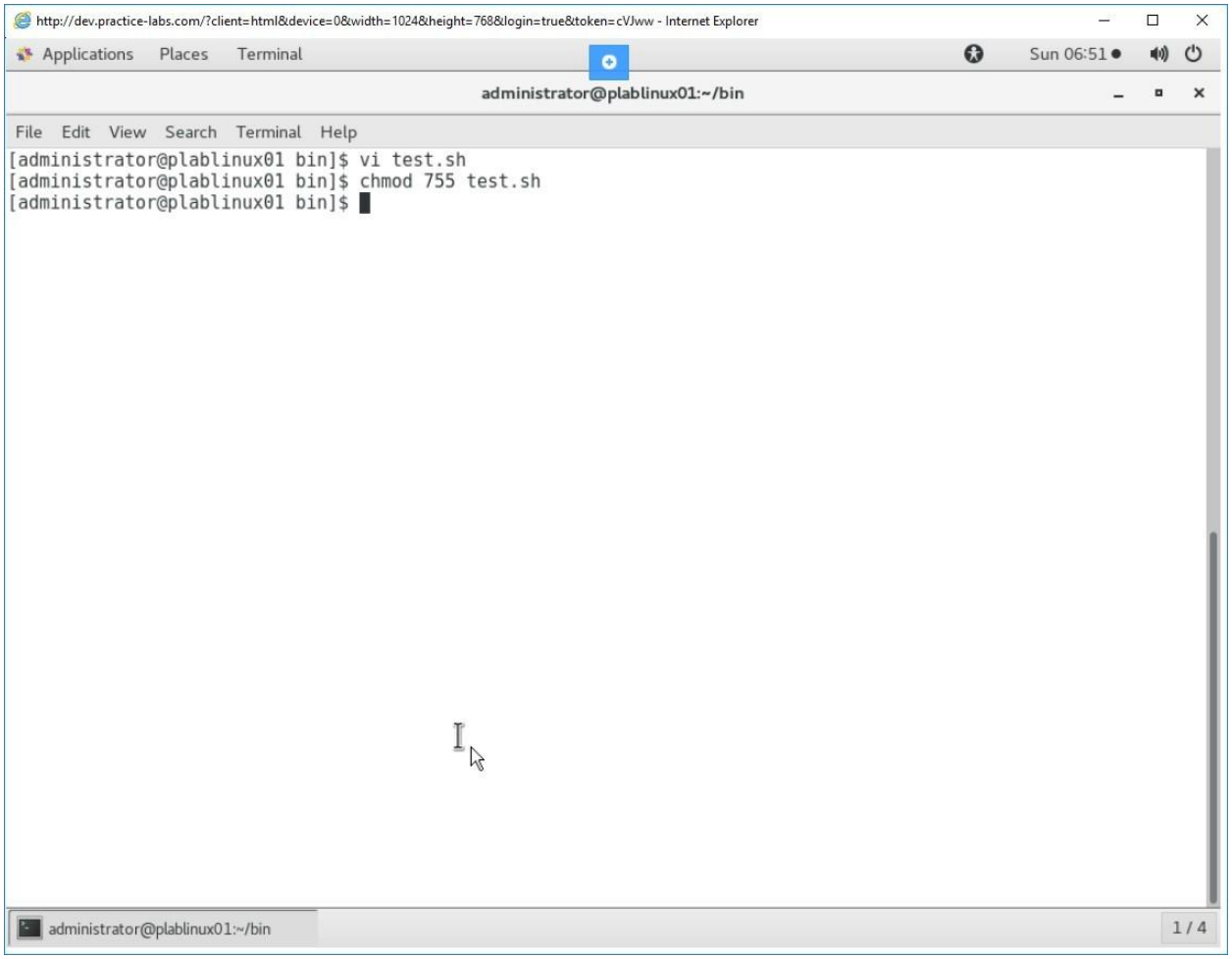


Figure 1.39 Screenshot of PLABLINUX01: Assigning the execute permission.

Step 3

Type the following command to execute the script:

```
test.sh
```

Press Enter. You are prompted for your name.

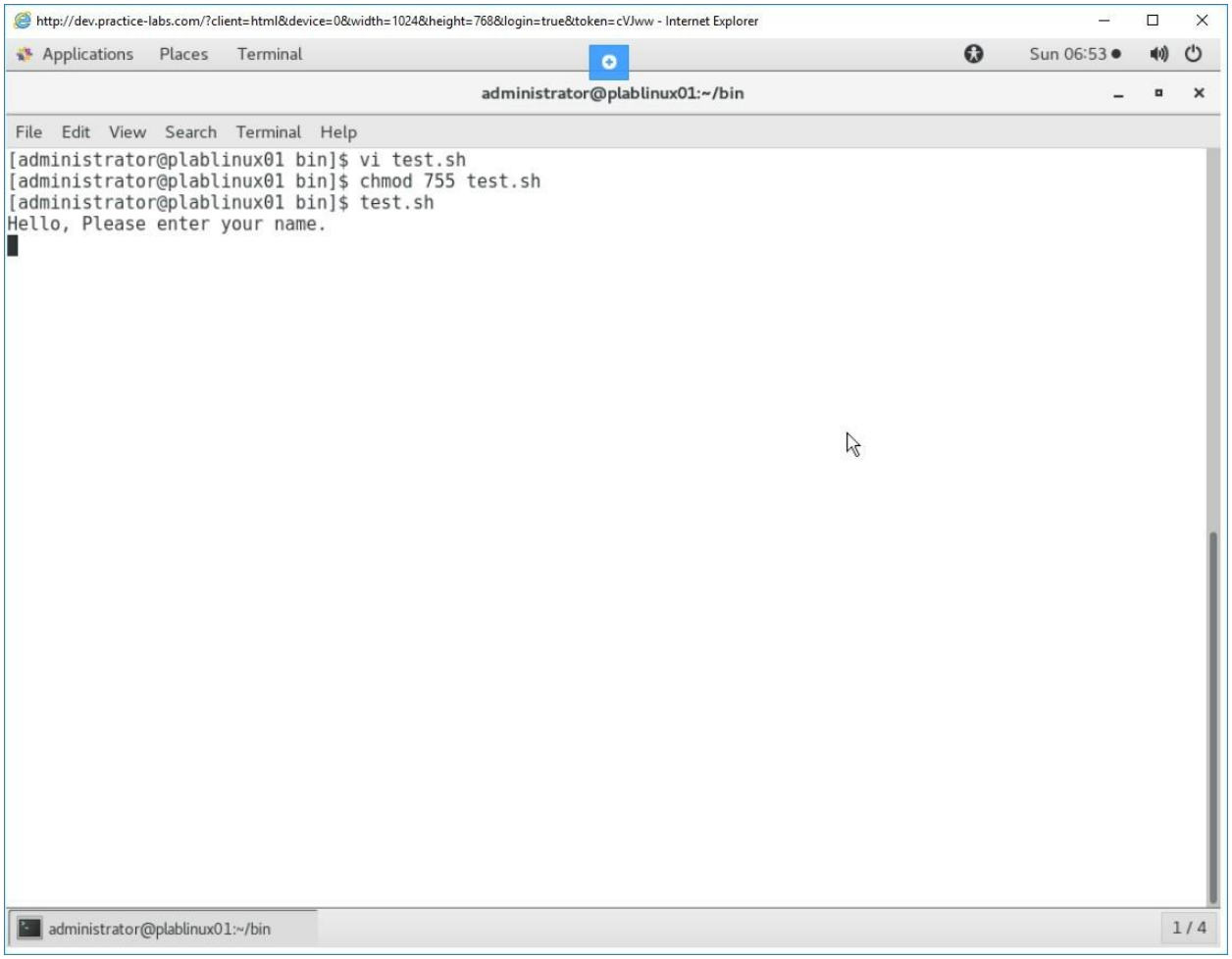


Figure 1:40 Screenshot of PLABLINUX01: Executing the shell script.

Step 4

Type your name and press Enter.

You are prompted with a welcome message.

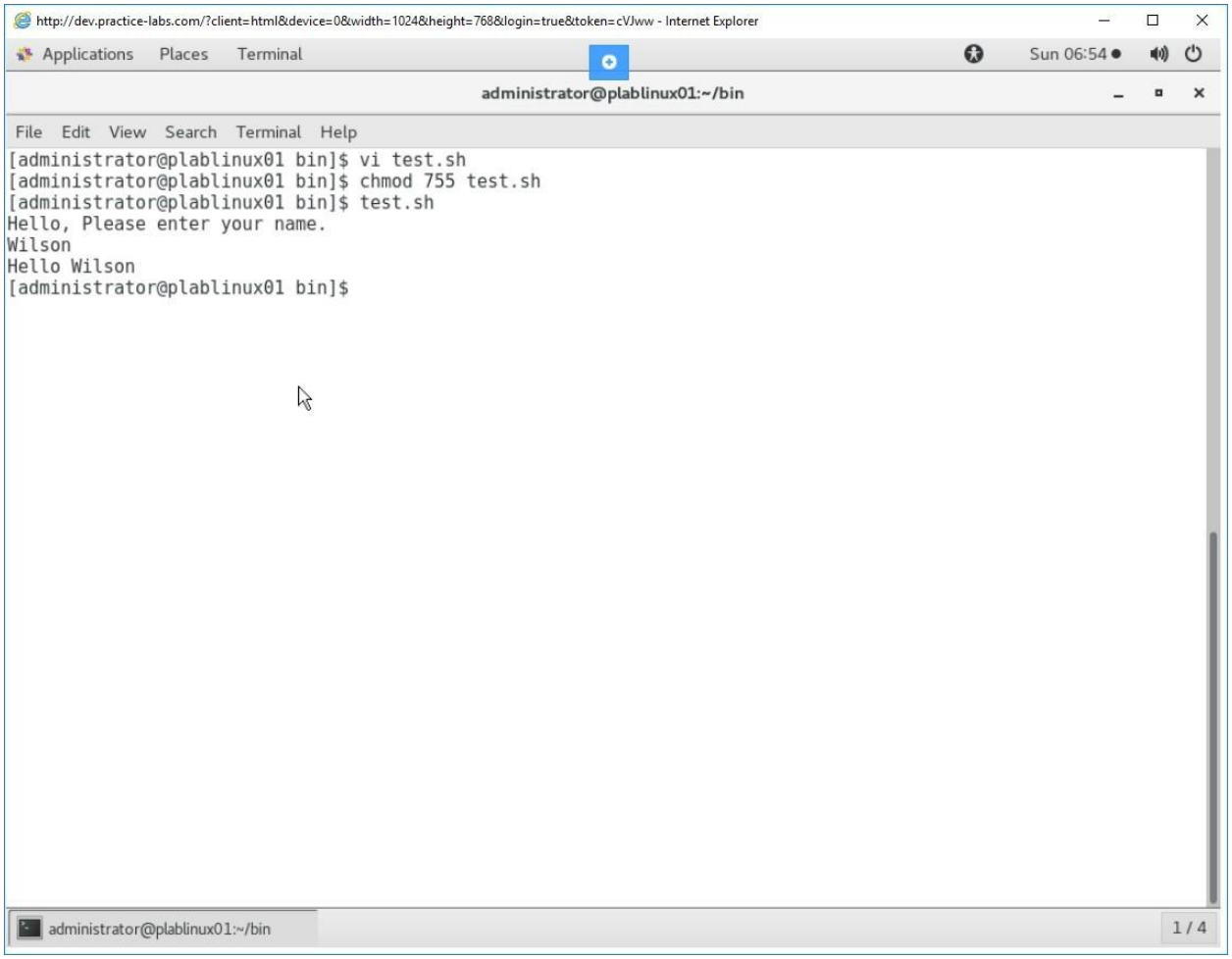


Figure 1.41 Screenshot of PLABLINUX01: Entering a value at the prompt.

Step 5

Create another script named fruits.sh. Use the following code:

```
#!/bin/bash
echo Name three fruits that you love to eat:
read fruit1 fruit2 fruit3
echo First fruit you like is: $fruit1
echo Second fruit you like is: $fruit2
echo Third fruit you like is: $fruit3
```

Save the file. In this script, you will accept multiple values using the read command. Read will split the values on whitespace.

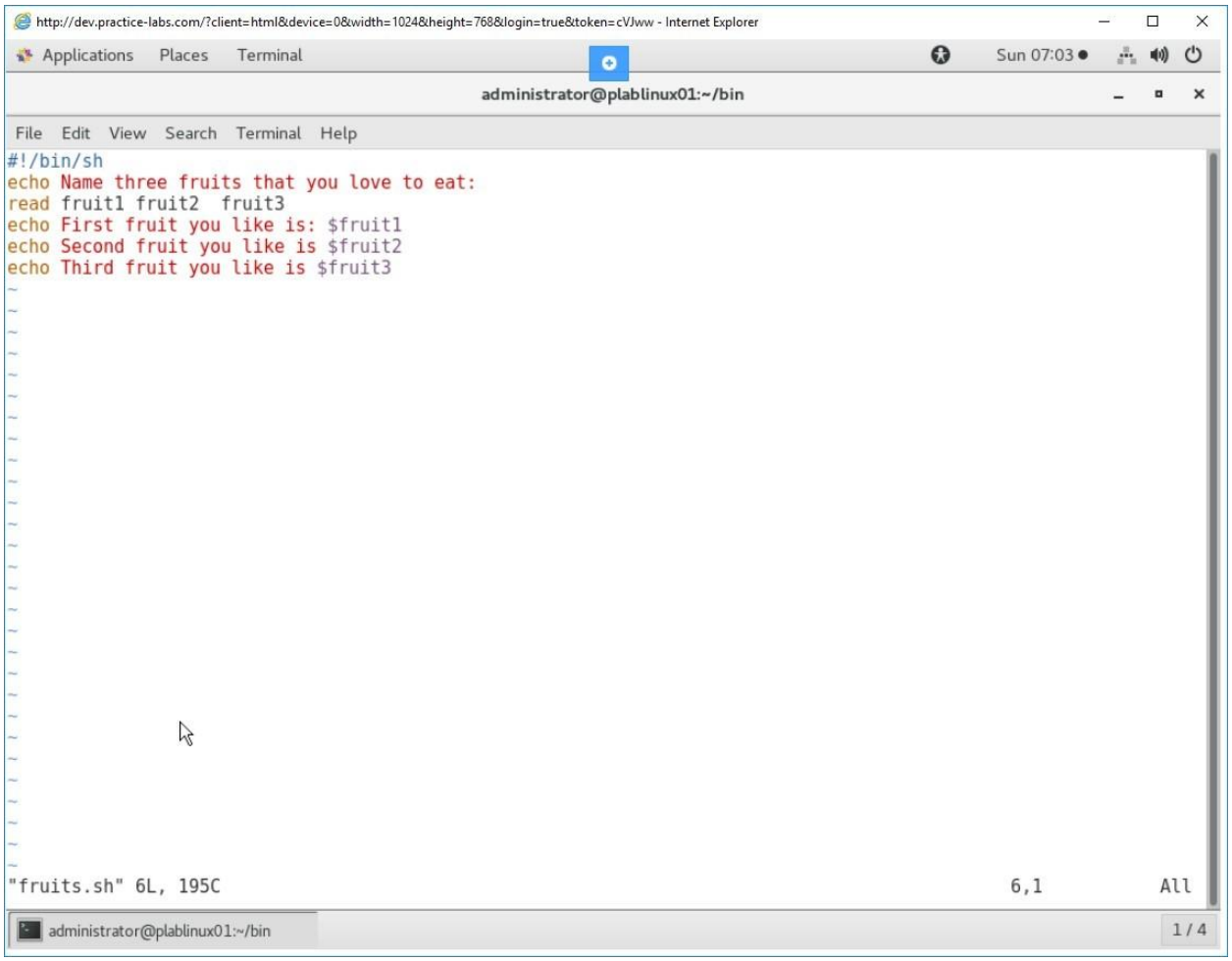


Figure 1.42 Screenshot of PLABLINUX01: Creating a new shell using the vi editor.

Step 6

After creating the file, assign the execute permissions.

Then, execute the script. Type the following command:

```
fruits.sh
```

Press Enter. Notice you are prompted to provide input.

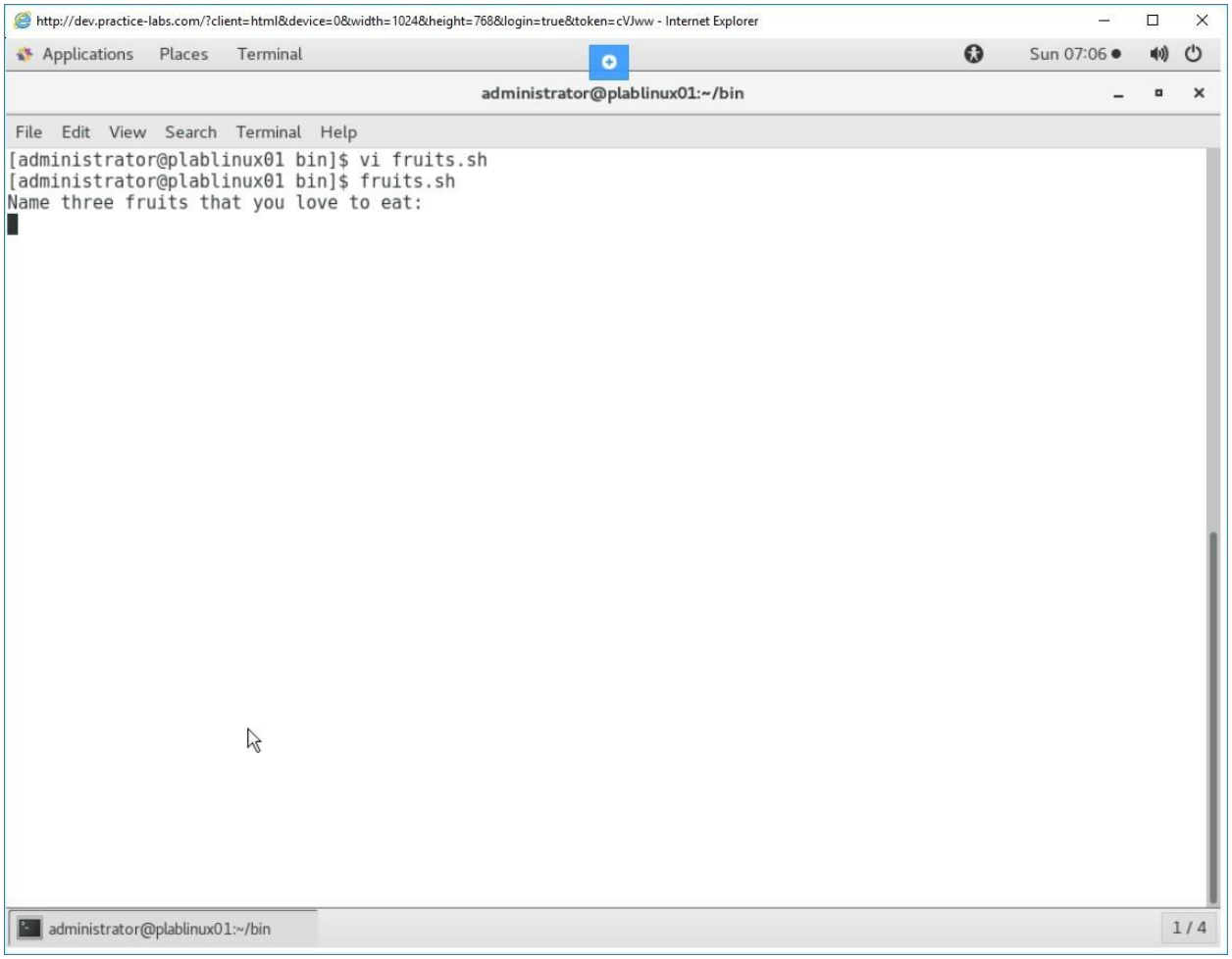


Figure 1.43 Screenshot of PLABLINUX01: Executing the shell script.

Step 7

Type the following values:

```
Apple  Orange Watermelon
```

Press Enter.

Note: You can use any values of your choice. They must be entered in the same sequence.

Notice that the entered values are assigned to the appropriate variables.

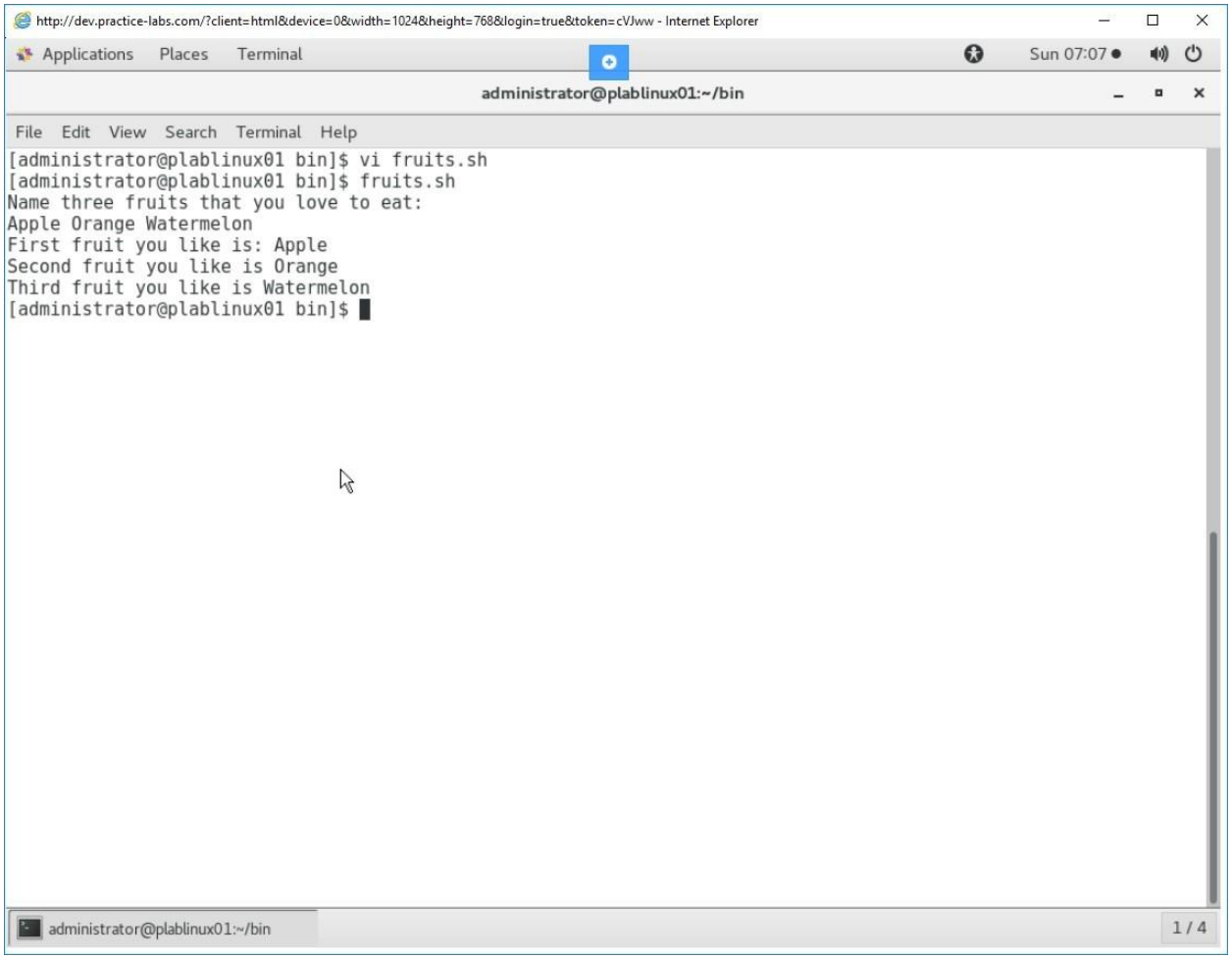


Figure 1.44 Screenshot of PLABLINUX01: Entering values at the prompt.

Keep all devices in their current state and proceed to the next exercise.

Review

Well done, you have completed the Working with Bash Profiles and Bash Scripts Practice Lab.

Summary

You completed the following exercise:

- Exercise 1 - Working with Bash Profiles and Bash Scripts

You should now be able to:

- Understanding the role of various bash related files
- Write a simple bash script
- Use commenting
- Use parameters
- Capture user inputs in scripts