**[Day 4 Task: Basic Linux Shell Scripting for DevOps Engineers.](https://github.com/DarshanaTakawale/90DaysOfDevOps/tree/master/2023/day04" \l "day-4-task-basic-linux-shell-scripting-for-devops-engineers)**

[**What is Kernel**](https://github.com/DarshanaTakawale/90DaysOfDevOps/tree/master/2023/day04#what-is-kernel)

The kernel is a computer program that is the core of a computer’s operating system, with complete control over everything in the system.

[**What is Shell**](https://github.com/DarshanaTakawale/90DaysOfDevOps/tree/master/2023/day04#what-is-shell)

A shell is special user program which provide an interface to user to use operating system services. Shell accept human readable commands from user and convert them into something which kernel can understand. It is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.

[**What is Linux Shell Scripting?**](https://github.com/DarshanaTakawale/90DaysOfDevOps/tree/master/2023/day04#what-is-linux-shell-scripting)

A shell script is a computer program designed to be run by a linux shell, a command-line interpreter. The various dialects of shell scripts are considered to be scripting languages. Typical operations performed by shell scripts include file manipulation, program execution, and printing text.

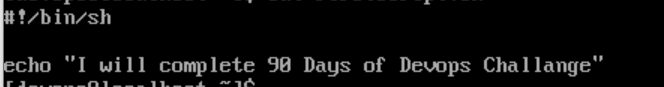
Tasks

**1) Explain in your own words and examples, what is Shell Scripting for DevOps.**

* Shell scripting is a programming language that is used for automating tasks and system administration in a Linux or Unix environment. In the context of DevOps, shell scripting plays a crucial role in automating repetitive tasks and streamlining the deployment and management of applications.
* With shell scripting, DevOps engineers can write scripts to automate tasks such as installing software packages, configuring systems, setting up network connections, and managing files and directories. This reduces the manual effort required to perform these tasks, which in turn saves time and improves efficiency.
* For example, suppose a DevOps engineer wants to automate the process of deploying an application to a server. With shell scripting, they can write a script that automates the entire process, including downloading the application code, setting up the necessary dependencies, configuring the server, and starting the application. This script can then be executed automatically or with a simple command, reducing the time and effort required for deployment.
* Furthermore, shell scripting allows DevOps engineers to create custom scripts that can be integrated with other DevOps tools such as CI/CD pipelines, monitoring and alerting systems, and configuration management tools. This integration enables the automation of end-to-end processes, from code development to deployment and monitoring.
* Overall, shell scripting is an essential skill for DevOps engineers as it enables the automation of complex and repetitive tasks, leading to increased efficiency and productivity in DevOps practices.
* Another example of this is If a QA team wants logs of a process performed the devops engineer will write a script to generate those logs using set of various text editing commands. And will provide its output to QA Team.
  1. **What is #!/bin/bash? can we write #!/bin/sh as well?**
* This first line (#!/bin/bash or #!/bin/sh) has a name. It is known as ‘she-bang‘(shabang). This derives from the concatenation of the tokens sharp (#) and bang (!). It is also called as sh-bang, hashbang, poundbang or hash-pling. In computing, a she-bang is the character sequence consisting of the characters number sign and exclamation mark (#!) at the beginning of a script.
* In this case, “/bin/bash” specifies the Bash shell as the interpreter. Bash is a popular shell on Linux and Unix systems and is commonly used for scripting.
* Yes, it is also possible to use “#!/bin/sh” in the shebang line. “/bin/sh” specifies the Bourne shell as the interpreter. The Bourne shell is an older shell that is less feature-rich than Bash, but it is still commonly available on many systems and can be used for simple scripting tasks.
* However, it is worth noting that Bash is backward compatible with Bourne shell syntax, so many scripts written for Bourne shell will work in Bash without modification. Therefore, unless there is a specific reason to use the Bourne shell, it is generally recommended to use Bash for shell scripting.
  + 1. **Write a Shell Script which prints I will complete #90DaysOofDevOps challenge**

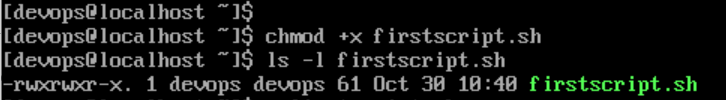
Here’s an example of a simple shell script that will print the message “I will complete #90DaysOfDevOps challenge” to the console:

This script uses the “echo” command to output the message to the console. The shebang line at the top of the script specifies that Bash should be used as the interpreter.



To run this script, you can save it to a file with a “.sh” extension (e.g. “firstscript.sh”), make it executable with the command “chmod +x firstscript.sh”, and then execute it with “./firstscript.sh” or by specifying the full path to the script.

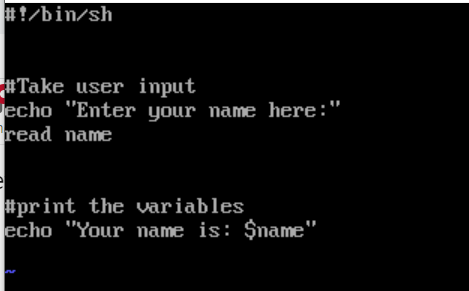


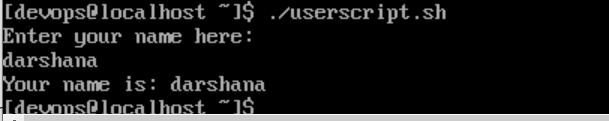




* + 1. **Write a Shell Script to take user input, input from arguments and print the variables.**

Here’s an example of a shell script that takes user input, input from arguments, and prints the variables:





This script prompts the user to enter their name, reads the input into the variable “name”.

* **Write an Example of If else in Shell Scripting by comparing 2 numbers**

Here’s an example of using if-else statements in shell scripting to compare two numbers: In this example, variables “num1” and “num2”, respectively. We then use if-else statements to compare these two numbers.