Unix Shell Scripting Tutorial for Beginners

Published On: October 10, 2024

Unix Shell Scripting Tutorial for Beginners

Utilizing a command-line shell is highly advantageous since it enables users to batch-run commands stored in a file. Learn the fundamentals in this Unix Shell Scripting tutorial designed for beginners.

Download Unix Shell Scripting Tutorial PDF

Introduction to Unix Shell Scripting

Even though Unix was created in the 1970s, it is still widely used today, particularly on servers and other powerful computers. We cover the following in this Unix Shell Scripting tutorial:

- Overview of Unix Shell Scripting
- Steps Involved with Shell Scripting
- Unix Shell Scripting Commands
- Shell Scripting Variables and Control Flow Statements
- Working with the First Unix Shell Script
- Advantages of Unix Shell Scripting

Overview of Unix Shell Scripting

Users can communicate with computers running

Unix through a command-line interface. They are prompted with a text window, where they can enter and carry out commands. Learn more in our **Unix Shell Scripting Course**.

UNIX was primarily known for its ease of use, huge software library, multitasking and multiuser capabilities, portability (ability to run on many different platforms), and hierarchical file system.

What is Unix Shell?

Unix Shell is the shell that runs on Unix-based operating systems like Linux, macOS, and others. Unlike "shell," which refers to any kind of command-line interface, "Unix shell" only refers to the shells found in Unix-based operating systems.

Numerous implementations of these Unix shells have different syntax and functionalities, such as file manipulation, process management, command execution, and input/output redirection.

Unix Shell Scripting Interview Questions

Types of Unix Shells

Many Unix shells are available with distinct capabilities to suit various user requirements; the most popular ones are as follows:

- **Bourne Shell (sh):** First released in 1970, the Bourne Shell (sh) is the first Unix shell. Basic scripting features including variables, read inputs, list files, and control flow statements are available.
- **Bash Shell (Bourne Again Shell):** Better scripting capabilities and other features like

MERN Stack Tutorial for Web Development Aspirants
Published On: October 14, 2024

MERN Stack Tutorial for Web Development Aspirants There is a growing need for competent MERN...

Unix Developer Salary in Chennai
Published On: October 12, 2024

Introduction A Unix Developer is responsible for creating, maintaining, and optimizing applications on Unix systems....

Tableau Developer Salary in Chennai
Published On: October 12, 2024

Introduction A Tableau Developer designs, develops, and maintains dashboards and visualizations using Tableau software. Key...

command-line editing are available in Bash, the enhanced version of Bourne Shell.

- **C Shell (csh):** This program is well-known for having interactive capabilities, including history substitution and a syntax similar to C.
- **Korn Shell (ksh):** It is an extension of Bourne Shell (sh) that offers more advanced scripting features and is backward compatible with Bourne Shell, created by David Korn.
- **Z Shell (zsh):** With sophisticated features like spelling correction, tab completion, and a plethora of setting choices, it's incredibly adaptable.
- **Fish (Friendly Interactive Shell):** fish (stylized in lowercase) is a Unix shell focused more on interactivity and usability with features like syntax highlighting, auto recommendations, and an intuitive scripting language.

Explore what our **Unix Shell Scripting course syllabus** has in store for your career.

Steps Involved with Shell Scripting

Unix Shell scripting is the term for shell scripting that is used exclusively in operating systems that are based on Unix. It is a specific subset of shell scripting carried out in a Unix setting.

The basic steps associated with shell scripting are

- Authoring the script
- Granting the shell access to it
- Allowing the shell to execute it

Terminologies of Shell Scripting

Understanding a few shell-related terms is crucial before delving deeper into shell programming, as it will help in our understanding:

- **Terminal:** A shell access interface. Users can issue commands and view the outcomes.
- **Shell Shebang Line:** An interpreter for running a shell script is specified in the first line of the script, which typically begins with #!.

VMware Tutorial for Cloud Computing Aspirants
Published On: October 12, 2024

VMware Tutorial for Cloud Computing Aspirants VMware software allows you to run a virtual machine...

    - **For example,** *#!/bin/bash* instructs the script to be executed by the Bash interpreter.
- **Shell Comments:** Lines that the shell interpreter skips over are called comments. They serve to clarify particular passages or the goal of the script.
    - Comments begin with a *# line*.
- **Shell Variables:** These are named variables that have values utilized in the script and are stored places.
    - The = operator can be used to set values for variables, and the variable name preceded by a dollar sign *(e.g., $name)* can be used to refer to the variables later in the script.
- **Commands:** Shell scripts are composed of lists of commands, which serve as their fundamental building elements.
    - These are built-in commands for external applications or built-in keywords *(e.g., cd to change directory, echo to print text)*.
- **Arguments:** Extra data that a script passes to commands are called arguments. They can be used to give precise instructions on how to utilize the command.
    - They begin with either -or — *(for example, git —version indicates the installed version of git)*.
- **Redirection:** This gives us control over the commands that the script receives and outputs.
    - For this, operators such as *>> (add to output), < (redirect input), and > (redirect output)* are utilized.
- **Control Flow Statements:** These specify the sequence in which the script is executed. Typical statements for control flows are as follows:
    - **If/then:** used to execute a command conditionally on the fulfillment of a true or false condition.
    - **For/while loops:** These loops are used to

repeat a set number of times (for loop) or until a predetermined condition is met (while loop).

Gain expertise with the computer fundamentals in our **C and C++ training in Chennai**.

Unix Shell Scripting Commands

We can use text commands to operate our computer thanks to the Unix shell. Here is a brief reference to a few key commands:

Navigate:

- **cd:** To move among directories (cd Desktop, for example).
- **ls:** To display every item in the chosen directory.

Files & Folders:

- **mkdir:** To create a new directory (folder).
- **touch:** To start a new, blank file.
- **rm:** To erase or remove a folder or file.
- **cp:** To use a terminal to copy files or folders.
- **mv:** To rename or move a file or folder.

Viewing & Text:

- **printf:** To display text on the computer screen.
- **cat:** To show the contents of a file on the terminal.
- **grep:** A program to look for text in files.

Permissions & Management:

- **chmod:** To add or modify permissions for a file or folder.
- **sudo:** To execute instructions as an administrator.

System Info:

- **df:** To view disk space use.
- **history:** To review previous orders.
- **ps:** To view the active processes.

Get placed in your dream job by enrolling in our **placement training institute in Chennai**.

Unix Shell Scripting Course Syllabus

Shell Scripting Variables and Control Flow Statements

The following is a summary of the essential elements that enable this:

Variables and User Input

Variables are designated storage spaces in shell scripting that can hold string, boolean, or numeric values.

- In Unix shell scripting, we can define a variable by writing its name and giving it a value.
- The read command can be used to extract variable values from user input.
- In a Unix shell, we can use the "$" symbol along with the variable name to access the value of any variable.

> **Example**
>
> *#!/bin/bash*
>
> *printf "What is your name? \n"*
>
> *read name*
>
> *printf "\n Hello, $name! \n"*

The user's name is utilized in the welcoming message of this script after being saved in the name variable.

Conditionals (if/else)

Conditional statements give you decision-making control in shell scripting. The "if/else" statements assess situations and carry out commands for the

results.

> **Syntax**
>
> *#!/bin/bash*
>
> *printf “Enter a number:”*
>
> *read number*
>
> *if [[“$number” -gt 10]]*
>
> *then*
>
> *printf “The number is greater than 10.”*
>
> *else*
>
> *printf “The number is not greater than 10.”*
>
> This script displays several messages based on whether the entered integer is bigger than 10.

Loops (for/while)

Loops provide effective processing for repeating operations in shell programming.

- The while loop continues iterating until a condition is no longer true, while the for loop iterates for a predetermined amount of times.
- These loops improve the functionality and adaptability of scripts by facilitating efficient automation and task execution.

> **For Loop Example**
>
> *#!/bin/bash*

```
for i in {1..5}

do

 echo “Number: $i”

done
```

This script uses a for loop that iterates five times to print the numbers 1 through 5.

Functions

In shell scripting, functions are the reusable code segments that are defined once and called repeatedly. In Unix shell scripts, a function can be defined and called using the following syntax:

```
#!/bin/bash

# Define a function to greet someone

welcome() {

 echo “Hello, $1!”

}

# Call the greet function with an argument

welcome “World”
```

A welcome function that accepts a name and shows a greeting is defined in this script. Next, the function is called passing in the input “World”.

Achieve your dream career with our wide range of **software training courses**.

Unix Developer Salary in Chennai

Working with the First Unix Shell Script "Hello World"

There are two ways we can run this Unix shell script, which are listed below.

Run scripts directly from the shell:

On a Unix-based operating system, such as MacOS or Linux, open the terminal or shell and type the command there, as indicated in the following image, to launch the Unix shell script directly:

printf 'Hello World'

Run the script using a script file:

This technique allows us to produce Unix shell script files, which typically finish in.sh (Bourne shell scripts),.bash (Bash scripts),.zsh (Z Shell),.ksh (Korn Shell), and.csh (C Shell).

We are going to develop a Bash shell script file in our example. The comprehensive instructions below are provided on constructing and running an executable bash script.

Step 1: Type the command below into your terminal to create a new, empty file called "helloworld.sh."

touch helloworld.sh

To create a new, empty file in the working directory of the terminal, type the touch command followed by the file name.

Step 2: Write the Unix shell script to output "Hello World!" after opening the created file in a text editor. For this example, the text editor Nano will be used.

To open the file in the nano editor, type the following command into the terminal.

nano helloworld.sh

When you run this command, your terminal will

launch a text editor. Now, type the following command as indicated below in the text editor:

#!/bin/sh

 # This is a bash program to display Hello World

printf "Hello World"

After that, hit "(ctrl + X)" to close the editor. You will be given the choice to save the file or not while you are leaving. To save the file, click "Yes."

Step 3: The Unix shell script file needs to be made executable.

We must change the file's permissions because, by default, the generated file cannot run as a program.

- To accomplish this, we must perform the command "chmod +x ," which adds the ability to run the particular file as a Unix script.
- To perform the command below in your terminal, browse to the directory where the file is placed and run "chmod +x "".

chmod +x helloworld.sh

Step 4: The last step is to use the command below in the terminal to execute the prepared script.

./helloworld.sh

Output

Advantages of Unix Shell Scripting

Here are the advantages of Unix Shell Scripting:

- **Automating Routine Operations:** System log rotation, user account creation, and file backups are just a few of the repetitious operations that software administrators frequently have to do.
- **System Administration and Configuration Management:** Complex configuration activities are automated with the use of scripts, which guarantee consistency and lower the possibility of human error.
- **Data Processing and Manipulation:** Text-based data processing and manipulation are areas in which shell scripting shines.
- **Customizing the User Experience:** By enabling the creation of unique commands and tools, shell scripting improves the user experience.
- **Development and Prototyping:** One useful tool for development and prototyping is shell scripting.

Conclusion

With this Unix Shell Scripting tutorial, we hope you have a fundamental understanding. Improve your Unix shell automation scripts through our **Unix shell scripting training in Chennai**.

Softlogic Systems

KK Nagar [Corporate Office]

No.10, PT Rajan Salai, K.K. Nagar, Chennai – 600 078.
Landmark: Karnataka Bank Building
Phone: +91 86818 84318
Email: enquiry@softlogicsys.in
Map: Google Maps Link

OMR

No. E1-A10, RTS Food Street
92, Rajiv Gandhi Salai (OMR),
Navalur, Chennai - 600 130.
Landmark: Adj. to AGS Cinemas
Phone: +91 89256 88858
Email: info@softlogicsys.in
Map: Google Maps Link

Courses

- Python
- Software Testing
- Full Stack Developer
- Java
- Power BI
- Clinical SAS
- Data Science
- Embedded
- Cloud Computing
- Hardware and Networking
- VBA Macros
- Mobile App Development
- DevOps

- Corporate Training
- Hire With Us
- Job Seekers
- SLA's Recently Placed Students
- Reviews
- Sitemap

Important Links

- Disclaimer
- Privacy Policy
- Terms and Conditions

Social Media Links

Review Sources

- Google
- Trustpilot
- Glassdoor
- Mouthshut
- Sulekha
- Justdial
- Ambitionbox
- Indeed
- Software Suggest
- Sitejabber