**What is shell Script?**

A shell script is a text [file](https://www.techtarget.com/whatis/definition/file) that contains a sequence of commands for a [UNIX](https://www.techtarget.com/searchdatacenter/definition/Unix)-based [operating system](https://www.techtarget.com/whatis/definition/operating-system-OS). It is called a shell script because it combines a sequence of commands, that would otherwise have to be typed into the keyboard one at a time, into a single script.

Shell scripting is a powerful tool commonly used across industries to automate tasks, test solutions, and increase efficiency.

**Who uses shell scripting?**

* **System administrators**:System administrators often use shell scripts for automating administrative tasks like backups, system monitoring, user account creation and management, and many more routine activities. This allows for efficiency, consistency, and accuracy.
* **Developers**: Developers often use shell scripting to automate development tasks, like automating file manipulation, deploying software to servers, running test suites, and more.
* **DevOps professionals**: In the world of DevOps, shell scripting can benefit tasks such as automation, configuration management, troubleshooting, and rapid iteration.

**Pros and cons of shell scripting    :**

Shell scripting has strengths and weaknesses, depending on your field and application.

* **Pros**

Shell scripting benefits professionals, especially system administrators, DevOps developers, and related professionals. As mentioned previously, it automates repetitive tasks and can boost productivity. It’s ideal for rapidly prototyping complex applications. Shell scripts are simple to create, modify, and debug. They’re also highly portable and capable of running on any UNIX-like system with minimal modifications.

* **Cons**

On the downside, shell scripts may be slower than programs written in compiled languages and less suited for complex computational tasks. Scripting languages may be more prone to costly errors than other programming languages.

**Examples of shell scripting**

Depending on your professional role, you might call on shell scripting for various functions. Here are a few scenarios in which you might choose to utilize the capabilities of shell scripting:

* **You have multiple databases on a single machine, or your requirements aren’t specific to a single database:**Shell scripting allows you to meet requirements that aren’t advisable or secure to fulfill using a single database.
* **You need to perform tasks when the database isn’t running:** In such cases, you can utilize a script to start or stop a database, as well as associated processes like listeners, which you can’t initiate from within the database itself.
* **You need to monitor the database’s status to ensure it’s running and ready to process queries:** A script can serve this purpose, monitoring not only the database but also other system processes and resources, providing a comprehensive view of the system’s operations.
* **You need to automate backups of your database:**For example, with Oracle's Recovery Manager (RMAN), you can create backup scripts that run on any platform. A shell script can call upon RMAN for a variety of backup and recovery activities.

Development:

**Variables**:

A variable used for storing data in programming. It includes a pointer to the memory location of the data as like other programming languages.

**Syntax**:

VariableName = VariableValue.

EX: Age = 25

**Access Variables**:

Ater declaring and assigning values to variables, you can access them using a dollar symbol ($) followed by the variable name.

Ex: echo $Age

Here echo is used for display the data.

Ex:

#!/bin/sh

AGE=25

echo $AGE

Output like below:



Assign values multiple times:

AGE=25

echo $AGE

AGE=35

echo $AGE

Output like below:



To restrict to assign multiple times:

AGE=25

echo $AGE

**readonly AGE**

AGE=35

echo $AGE



Types of Variables:

Local Variables

Global Variables

Local Variables:

#!/bin/sh

setAge() {

address=Nellore

echo "$address"

echo "Inside Function Age: $AGE"

}

AGE=40

setAge

echo "Script Age: $tmp"

Output:

