

Unix / Linux - Using Shell Variables

In this chapter, we will learn how to use Shell variables in Unix. A variable is a character string to which we assign a value. The value assigned could be a number, text, filename, device, or any other type of data.

A variable is nothing more than a pointer to the actual data. The shell enables you to create, assign, and delete variables.

Variable Names

The name of a variable can contain only letters (a to z or A to Z), numbers (0 to 9) or the underscore character (_).

By convention, Unix shell variables will have their names in UPPERCASE.

The following examples are valid variable names –

```
_ALI  
TOKEN_A  
VAR_1  
VAR_2
```

Following are the examples of invalid variable names –

```
2_VAR  
-VARIABLE  
VAR1-VAR2  
VAR_A!
```

The reason you cannot use other characters such as !, *, or - is that these characters have a special meaning for the shell.

Defining Variables

Variables are defined as follows –

```
variable_name=variable_value
```

For example –

```
NAME="Zara Ali"
```

The above example defines the variable `NAME` and assigns the value "Zara Ali" to it. Variables of this type are called **scalar variables**. A scalar variable can hold only one value at a time.

Shell enables you to store any value you want in a variable. For example –

```
VAR1="Zara Ali"  
VAR2=100
```

Accessing Values

To access the value stored in a variable, prefix its name with the dollar sign (\$) –

For example, the following script will access the value of defined variable `NAME` and print it on STDOUT –

[Live Demo](#)

```
#!/bin/sh  
  
NAME="Zara Ali"  
echo $NAME
```

The above script will produce the following value –

```
Zara Ali
```

Read-only Variables

Shell provides a way to mark variables as read-only by using the read-only command. After a variable is marked read-only, its value cannot be changed.

For example, the following script generates an error while trying to change the value of `NAME` –

[Live Demo](#)

```
#!/bin/sh  
  
NAME="Zara Ali"  
readonly NAME  
NAME="Qadiri"
```

The above script will generate the following result –

```
/bin/sh: NAME: This variable is read only.
```

Unsetting Variables

Unsetting or deleting a variable directs the shell to remove the variable from the list of variables that it tracks. Once you unset a variable, you cannot access the stored value in the variable.

Following is the syntax to unset a defined variable using the **unset** command –

```
unset variable_name
```

The above command unsets the value of a defined variable. Here is a simple example that demonstrates how the command works –

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

```
NAME="Zara Ali"
```

```
unset NAME
```

```
echo $NAME
```

The above example does not print anything. You cannot use the unset command to **unset** variables that are marked **readonly**.

Variable Types

When a shell is running, three main types of variables are present –

- **Local Variables** – A local variable is a variable that is present within the current instance of the shell. It is not available to programs that are started by the shell. They are set at the command prompt.
 - **Environment Variables** – An environment variable is available to any child process of the shell. Some programs need environment variables in order to function correctly. Usually, a shell script defines only those environment variables that are needed by the programs that it runs.
 - **Shell Variables** – A shell variable is a special variable that is set by the shell and is required by the shell in order to function correctly. Some of these variables are environment variables whereas others are local variables.
-
-