How to Create a Shell Script in Linux

Shell is an interface of the operating system. It accepts commands from users and interprets them to the operating system. If you want to run a bunch of commands together, you can do so by creating a shell script. Shell scripts are very useful if you need to do a task routinely, like taking a backup. You can list those commands and execute them all with just a single script. Let's see how you can create a shell script and run it on Linux.

Creating a Shell Script

Login to your Linux machine and open the terminal, navigate to the folder where you want to store the shell script. Shell scripts end with the extension ".sh". Let's create our first shell script. Type in

```
touch script.sh
```

Now, this script file is not executable by default, we have to give the executable permission to this file. Type in

```
chmod +x script.sh
```

Now, we will add some commands to this shell script. Open this shell script with any text editor of your choice (command-line based or GUI based) and add some commands. We will use nano. Type in

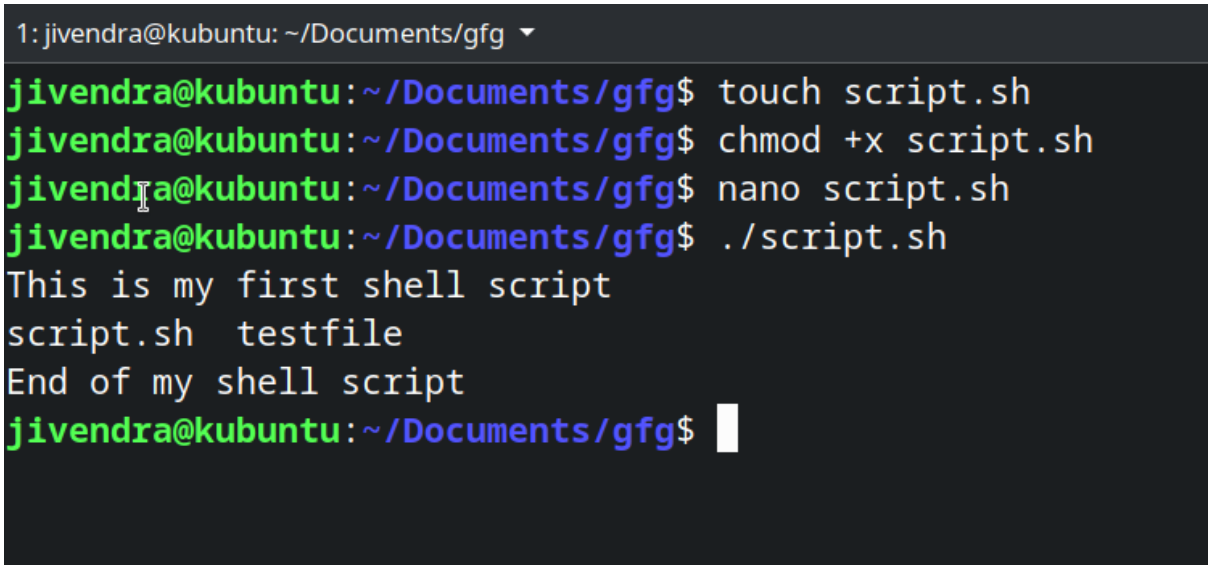
```
nano script.sh
```

Add the following commands to test this shell script

```
echo This is my first shell script
touch testfile
ls
echo End of my shell script
```

Save the changes, and run the shell script by typing in

```
./script.sh
```



Screenshot of above steps

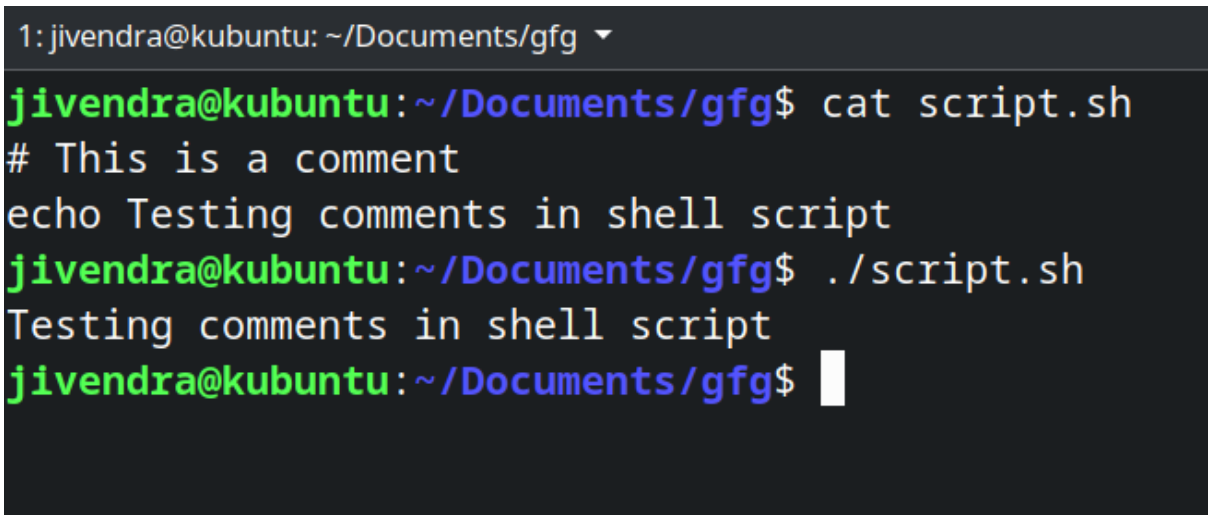
You can see, it executed all the specified commands.

Comments in the shell script

Any line which starts with "#" in the shell script is treated as a comment and is ignored by the shell during execution, except the shebang line, which we will see later in this article. Let's see an example. A shell script is created with the following content.

```
# This is a comment

echo Testing comments in shell script
```



Comments in Shell Script

You can see, the comment gets ignored.

Variables in Shell Script

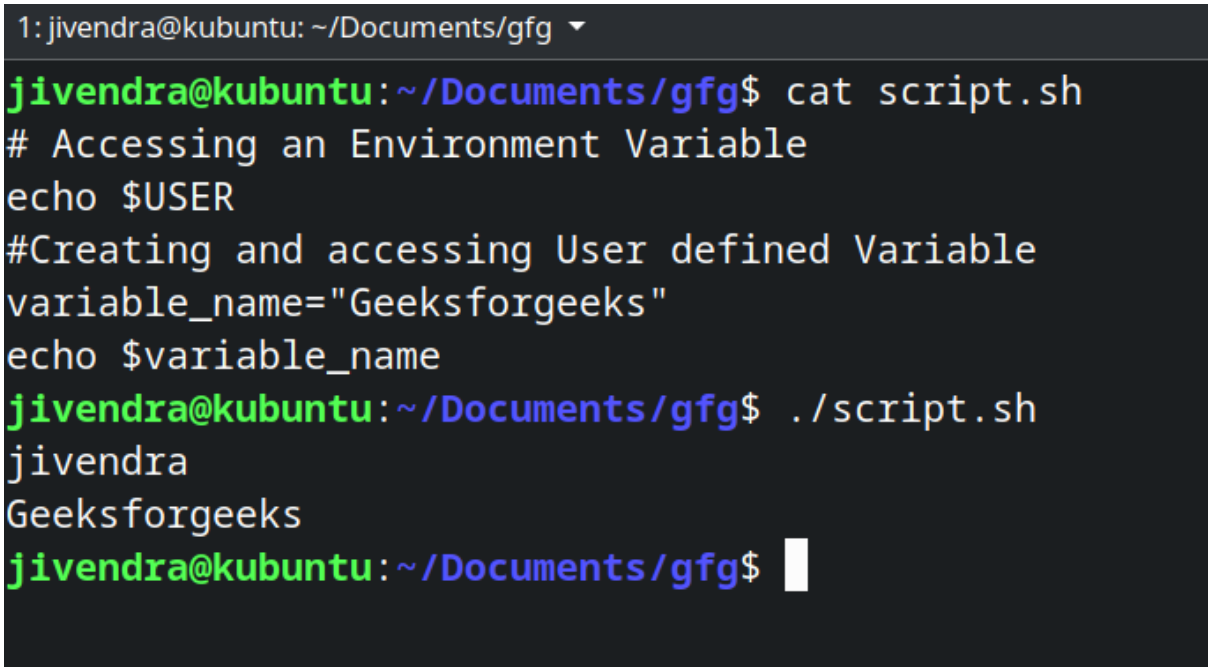
Yes, Shell scripts support the use of variables, and we need not define a variable's type during its declaration. There are two types of variables:

- System Defined variables
- User-Defined Variables.

System-defined variables, also called environment variables, are generally Capitalised. You can view all the current environment variables using the printenv command. User-Defined variables are set by the user, and they exist only during script execution. You can define a variable by simply typing its name and assigning a value with = sign and access a variable by adding a $ before the variable name. Variables are demonstrated in the following example script.

```
# Accessing an Environment Variable
echo $USER


# Creating and accessing User defined Variable
variable_name="Geeksforgeeks"
echo $variable_name
```



Variables in Shell Script