

Introduction on Shell Scripting

1. What is a Shell?

An operating system consists of many components, but the two prime components are Kernel and Shell, as in Figure 1. A Kernel is at the nucleus of a computer. It makes the communication between the hardware and software possible. While the Kernel is the innermost part of an operating system, a shell is the outermost one. A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.

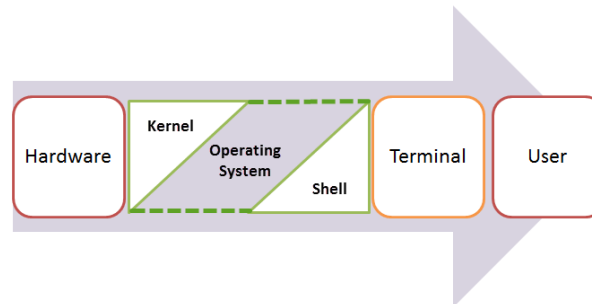


Figure 1. Operating system with Kernel and Shell

When you run the terminal, the Shell issues a **command prompt (usually \$)**, where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal. The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name **Shell**.

2. What is Shell Scripting?

Shell scripting is writing a series of command for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user.

Let us understand the steps in creating a shell script

1. **Create a file using a vi editor** (or any other editor). Name script file with **extension .sh**
2. **Start** the script with **#!/bin/sh**
The “#!” is an operator called shebang which directs the script to the interpreter location. So, if we use “#!/bin/sh” the script gets directed to the shell.
3. Write some code.
4. Save the script file as filename.sh
5. For **executing** the script type **./filename.sh**

Let’s see a simple example as below:

```
#!/bin/sh
pwd
ls
```

The execution results are presented in Figure 2:

```
xuelin@xuelin-GTX970: /lib
xuelin@xuelin-GTX970:/lib$ ./scriptsample.sh
/lib
apparmor      klibc-k3La8MUuzHQ0_kG8hokcGAC0PA.so  scriptsample.sh
brltty        ld-linux.so.2                        systemd
cpp           linux-sound-base                     sysvinit
crda          lsb                                  terminfo
firmware      modprobe.d                           udev
hdparm        modules                              ufw
i386-linux-gnu nvidia-384                           x86_64-linux-gnu
lftpdown      recovery-mode                        xtables
lnst          resolvconf
xuelin@xuelin-GTX970:/lib$
```

Figure 2. A simple shell script

3. Most frequently used Shell commands

When writing Shell scripts, we have commands used frequently which help us perform job automations more effectively and efficiently. For example, the **ls** command used in Section 2 show all the major directories and files in a given file system. The command is used for viewing files, folders, and directories. Also, the **pwd** is the command to display the current directory path.

The pages listed below contain more informative details which will give you a clearer picture of those commonly used commands in Shell.

The 10 Most Important Linux Commands:

<http://www.informit.com/blogs/blog.aspx?uk=The-10-Most-Important-Linux-Commands>

50 Most Frequently Used UNIX/Linux Commands (With Examples)

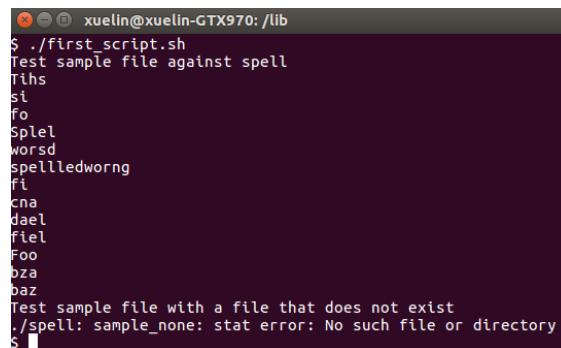
https://www.thegeekstuff.com/2010/11/50-linux-commands/?utm_source=feedburner

4. A simple example for Project 1

Below you will find a simple example of how to write your script for Project 1. Suppose you are in the folder of your project files with the sample application. The code is as follows:

```
#!/bin/sh
echo "Test sample file against spell"
./spell sample
echo "Test sample file with a file that
does not exist"
./spell sample_none
```

The execution result is presented in Figure 3.

A terminal window with a dark background and light text. The window title is 'xuelin@xuelin-GTX970: /lib'. The prompt is '\$./first_script.sh'. The output shows the script's execution: 'Test sample file against spell' followed by a list of words (Tih, si, fo, Spl, word, spelled, wrong, fi, cna, dael, fiel, Foo, oza, baz). Then, it shows 'Test sample file with a file that does not exist' followed by an error message: './spell: sample_none: stat error: No such file or directory'. The prompt '\$' is visible at the bottom.

```
xuelin@xuelin-GTX970: /lib
$ ./first_script.sh
Test sample file against spell
Tih
si
fo
Spl
word
spelled
wrong
fi
cna
dael
fiel
Foo
oza
baz
Test sample file with a file that does not exist
./spell: sample_none: stat error: No such file or directory
$
```

Figure 3. Execution of a simple example for Project 1