In Bash, each command or command sequence is delimited by using a semicolon or a new line. For example:

```
$ cmd1 ; cmd2
```

This is equivalent to:

```
$ cmd1
$ cmd2
```

Finally, the # character is used to denote the beginning of unprocessed comments. A comment section starts with # and proceeds up to the end of that line. The comment lines are most often used to provide comments about the code in the file or to stop a line of code from being executed.

Now let us move on to the basic recipes in this chapter.

Printing in the terminal

The **terminal** is an interactive utility by which a user interacts with the shell environment. Printing text in the terminal is a basic task that most shell scripts and utilities need to perform regularly. As we will see in this recipe, this can be performed via various methods and in different formats.

How to do it...

`echo` is the basic command for printing in the terminal.

`echo` puts a newline at the end of every echo invocation by default:

```
$ echo "Welcome to Bash"
Welcome to Bash
```

Simply, using double-quoted text with the `echo` command prints the text in the terminal. Similarly, text without double quotes also gives the same output:

```
$ echo Welcome to Bash
Welcome to Bash
```

Another way to do the same task is by using single quotes:

```
$ echo 'text in quotes'
```

These methods may look similar, but some of them have a specific purpose and side effects too. Consider the following command:

```
$ echo "cannot include exclamation - ! within double quotes"
```

This will return the following output:

```
bash: !: event not found error
```

Hence, if you want to print special characters such as `!`, either do not use them within double quotes or escape them with a special escape character (`\`) prefixed with it, like so:

```
$ echo Hello world !
```

Or:

```
$ echo 'Hello world !'
```

Or:

```
$ echo "Hello world \!" #Escape character \ prefixed.
```

The side effects of each of the methods are as follows:

- When using `echo` without quotes, we cannot use a semicolon, as it acts as a delimiter between commands in the Bash shell
- `echo hello; hello` takes `echo hello` as one command and the second `hello` as the second command
- Variable substitution, which is discussed in the next recipe, will not work within single quotes

Another command for printing in the terminal is `printf`. It uses the same arguments as the `printf` command in the C programming language. For example:

```
$ printf "Hello world"
```

`printf` takes quoted text or arguments delimited by spaces. We can use formatted strings with `printf`. We can specify string width, left or right alignment, and so on. By default, `printf` does not have newline as in the `echo` command. We have to specify a newline when required, as shown in the following script:

```
#!/bin/bash
#Filename: printf.sh

printf  "%-5s %-10s %-4s\n" No Name  Mark
printf  "%-5s %-10s %-4.2f\n" 1 Sarath 80.3456
printf  "%-5s %-10s %-4.2f\n" 2 James 90.9989
printf  "%-5s %-10s %-4.2f\n" 3 Jeff 77.564
```