How to do it...

We can create functions to perform tasks and we can also create functions that take parameters (also called arguments) as you can see in the following steps:

1. A function can be defined as follows:

```
function fname()
{
    statements;
}
Or alternately,
fname()
{
    statements;
}
```

2. A function can be invoked just by using its name:

```
$ fname ; # executes function
```

3. Arguments can be passed to functions and can be accessed by our script:

```
fname arg1 arg2 ; # passing args
```

   Following is the definition of the function `fname`. In the `fname` function, we have included various ways of accessing the function arguments.

```
fname()
{
  echo $1, $2; #Accessing arg1 and arg2
  echo "$@"; # Printing all arguments as list at once
  echo "$*"; # Similar to $@, but arguments taken as single entity
  return 0; # Return value
}
```

   Similarly, arguments can be passed to scripts and can be accessed by `script:$0` (the name of the script):

   - `$1` is the first argument
   - `$2` is the second argument
   - `$n` is the *n*th argument
   - `"$@"`expands as `"$1" "$2" "$3"` and so on
   - `"$*"` expands as `"$1c$2c$3"`, where `c` is the first character of IFS
   - `"$@"` is used more often than `"$*"`since the former provides all arguments as a single string

There's more...

Let us explore through more tips on Bash functions.

The recursive function

Functions in Bash also support recursion (the function that can call itself). For example, `F() { echo $1; F hello; sleep 1; }`.

Fork bomb

We can write a recursive function, which is basically a function that calls itself:

```
:(){ :|:& };:
```

It infinitely spawns processes and ends up in a denial-of-service attack. `&` is postfixed with the function call to bring the subprocess into the background. This is a dangerous code as it forks processes and, therefore, it is called a fork bomb.

You may find it difficult to interpret the preceding code. See the Wikipedia page `http://en.wikipedia.org/wiki/Fork_bomb` for more details and interpretation of the fork bomb.

It can be prevented by restricting the maximum number of processes that can be spawned from the `config` file at `/etc/security/limits.conf`.

Exporting functions

A function can be exported—like environment variables—using `export`, such that the scope of the function can be extended to subprocesses, as follows:

```
export -f fname
```

Reading the return value (status) of a command

We can get the return value of a command or function in the following way:

```
cmd;
echo $?;
```

`$?` will give the return value of the command `cmd`.

The return value is called **exit status**. It can be used to analyze whether a command completed its execution successfully or unsuccessfully. If the command exits successfully, the exit status will be zero, otherwise it will be a nonzero value.