The character offset for a string in a line is a counter from `0`, starting with the first character. In the preceding example, `not` is at the seventh offset position (that is, not starts from the seventh character in the line; that is, `gnu is not unix`).

The `-b` option is always used with `-o`.

12. To search over multiple files, and list which files contain the pattern, we use the following:

```
$ grep -l linux sample1.txt sample2.txt
sample1.txt
sample2.txt
```

The inverse of the `-l` argument is `-L`. The `-L` argument returns a list of non-matching files.

There's more...

We have seen the basic usages of the `grep` command, but that's not it; the `grep` command comes with even more features. Let's go through those.

Recursively search many files

To recursively search for a text over many directories of descendants, use the following command:

```
$ grep "text" . -R -n
```

In this command, "`.`" specifies the current directory.

The options `-R` and `-r` mean the same thing when used with `grep`.

For example:

```
$ cd src_dir
$ grep "test_function()" . -R -n
./miscutils/test.c:16:test_function();
```

`test_function()` exists in line number 16 of `miscutils/test.c`.

This is one of the most frequently used commands by developers. It is used to find files in the source code where a certain text exists.

Ignoring case of pattern

The `-i` argument helps match patterns to be evaluated, without considering the uppercase or lowercase. For example:

```
$ echo hello world | grep -i "HELLO"
hello
```

grep by matching multiple patterns

Usually, we specify single patterns for matching. However, we can use an argument `-e` to specify multiple patterns for matching, as follows:

```
$ grep -e "pattern1" -e "pattern"
```

This will print the lines that contain either of the patterns and output one line for each match. For example:

```
$ echo this is a line of text | grep -e "this" -e "line" -o
this
line
```

There is also another way to specify multiple patterns. We can use a pattern file for reading patterns. Write patterns to match line-by-line, and execute `grep` with a `-f` argument as follows:

```
$ grep -f pattern_filesource_filename
```

For example:

```
$ cat pat_file
hello
cool

$ echo hello this is cool | grep -f pat_file
hello this is cool
```

Including and excluding files in a grep search

`grep` can include or exclude files in which to search. We can specify `include` files or `exclude` files by using wild card patterns.

To search only for `.c` and `.cpp` files recursively in a directory by excluding all other file types, use the following command:

```
$ grep "main()" . -r  --include *.{c,cpp}
```