Getting ready

The `sort` command accepts input as filenames, as well as from `stdin` (standard input) and outputs the result by writing into `stdout`. The same applies to the `uniq` command.

How to do it...

1. We can easily sort a given set of files (for example, `file1.txt` and `file2.txt`) as follows:

   ```
   $ sort file1.txt file2.txt > sorted.txt
   ```

   Or:

   ```
   $ sort file1.txt file2.txt -o sorted.txt
   ```

2. For a numerical sort, we can use:

   ```
   $ sort -n file.txt
   ```

3. To sort in the reverse order, we can use:

   ```
   $ sort -r file.txt
   ```

4. For sorting by months (in the order Jan, Feb, March,...), use:

   ```
   $ sort -M months.txt
   ```

5. To merge two already sorted files, use:

   ```
   $ sort -m sorted1 sorted2
   ```

6. To find the unique lines from a sorted file, use:

   ```
   $ sort file1.txt file2.txt | uniq
   ```

7. To check if a file has already been sorted, use:

   ```
   #!/bin/bash
   #Desc: Sort
   sort -C filename ;
   if [ $? -eq 0 ]; then
      echo Sorted;
   else
      echo Unsorted;
   fi
   ```

   Replace `filename` with the file you want to check and run the script.

How it works...

As shown in the examples, `sort` takes numerous parameters that can be used to sort the data in files in different ways. Furthermore, it is useful when using the `uniq` command, which expects its input to be sorted.

There are numerous scenarios where the `sort` and `uniq` commands can be used. Let's go through the various options and usage techniques.

For checking if a file is already sorted or not, we exploit the fact that `sort` returns an exit code (`$?`) of 0 if the file is sorted and nonzero otherwise.

There's more...

These were some basic usages of the `sort` command. Let us see some ways of using it to accomplish complex tasks:

Sorting according to the keys or columns

We can use a column with sort if we need to sort a text as follows:

```
$ cat data.txt
1  mac    2000
2  winxp    4000
3  bsd    1000
4  linux    1000
```

We can sort this in many ways; currently it is numeric, sorted by the serial number (the first column). We can also sort by the second column and the third column.

`-k` specifies the key by which the sort is to be performed. Key is the column number by which sort is to be done. `-r` specifies the sort command to sort in the reverse order. For example:

```
# Sort reverse by column1
$ sort -nrk 1  data.txt
4  linux    1000
3  bsd    1000
2  winxp    4000
1  mac    2000
# -nr means numeric and reverse

# Sort by column 2
$ sort -k 2  data.txt
```