To find duplicate lines in the file:

```
$ sort unsorted.txt  | uniq -d
hack
```

To specify keys, we can use the combination of the `-s` and `-w` arguments.

- `-s` specifies the number for the first *N* characters to be skipped
- `-w` specifies the maximum number of characters to be compared

This comparison key is used as the index for the `uniq` operation as follows:

```
$ cat data.txt
u:01:gnu
d:04:linux
u:01:bash
u:01:hack
```

We need to use the highlighted characters as the unique key. This is used to ignore the first two characters (`-s 2`) and the maximum number of comparison characters is specified using the `-w` option (`-w 2`):

```
$ sort data.txt | uniq -s 2 -w 2
d:04:linux
u:01:bash
```

While we use output from one command as input to the `xargs` command, it is always preferable to use a zero-byte terminator for each of the lines of the output, which act as the source for `xargs`. While using the `uniq` commands output as the source for `xargs`, we should use a zero terminated output. If a zero-byte terminator is not used, by default the space characters are used as the delimiter to split the arguments in the `xargs` command. For example, a line with text "this is a line" from `stdin` will be taken as four separate arguments by the `xargs` command. Actually, it is a single line. When a zero-byte terminator is used, `\0` is used as the delimiter character and, hence, a single line including a space is interpreted as a single argument.

Zero-byte-terminated output can be generated from the `uniq` command as follows:

```
$ uniq -z file.txt
```

The following command removes all the files, with filenames read from `files.txt`:

```
$ uniq -z file.txt | xargs -0 rm
```

If multiple-line entries of filenames exist in the file, the `uniq` command writes the filename only once to `stdout`.

Temporary file naming and random numbers

While writing shell scripts, we often need to store temporary data. The most suitable location to store temporary data is `/tmp` (which will be cleaned out by the system on reboot). We can use two methods to generate standard filenames for temporary data.

How to do it...

Perform the following steps to create a temporary file and perform different naming operations on it:

1. Create a temporary file as follows:

```
$ filename=`mktemp`
$ echo $filename
/tmp/tmp.8xvhkjF5fH
```

   This will create a temporary file and print its filename which we store in `$filename` in this example.

2. To create a temporary directory, use the following commands:

```
$ dirname=`mktemp -d`
$ echo $dirname
tmp.NI8xzW7VRX
```

   This will create a temporary directory and print its filename which we store in `$dirname` in this example.

3. To just generate a filename without actually creating a file or directory, use this:

```
$ tmpfile=`mktemp -u`
$ echo $tmpfile
/tmp/tmp.RsGmilRpcT
```

   Here, the filename will be stored in `$tmpfile`, but the file won't be created.

4. To create the temporary filename according to a template, use:

```
$mktemp test.XXX
test.2tc
```