See also

- The *Using awk for advanced text processing* recipe in this chapter explains the `awk` command

Printing lines in the reverse order

This is a very simple recipe. It may not seem very useful, but it can be used to emulate the stack datastructure in Bash. This is something interesting. Let's print the lines of text in a file in reverse order.

Getting ready

A little hack with `awk` can do the task. However, there is a direct command, `tac`, to do the same as well. `tac` is the reverse of `cat`.

How to do it...

We will first see how to do this with `tac`.

1. The `tac` syntax is as follows:

   ```
   tac file1 file2 …
   ```

   It can also read from `stdin`, as follows:

   ```
   $ seq 5 | tac
   5
   4
   3
   2
   1
   ```

   In `tac`, `\n` is the line separator. But, we can also specify our own separator by using the `-s` "separator" option.

2. We can do it in `awk` as follows:

   ```
   $ seq 9 | \
   awk '{ lifo[NR]=$0 }
   END{ for(lno=NR;lno>-1;lno--){ print lifo[lno]; }
   }'
   ```

   `\` in the shell script is used to conveniently break a single line command sequence into multiple lines.

How it works...

The `awk` script is very simple. We store each of the lines into an associative array with the line number as an array index (`NR` returns the line number). In the end, `awk` executes the `END` block. In order to get the last line number, `lno=NR` is used in the `{ }` block. Hence, it iterates from the last line number to `0`, and prints the lines stored in the array in reverse order.

Parsing e-mail addresses and URLs from text

Parsing a required text from a given file is a common task that we encounter in text processing. Items such as, e-mails and URLs can be found out with the help of correct regex sequences. Mostly, we need to parse e-mail addresses from a contact list of an e-mail client, which is composed of many unwanted characters and words, or from an HTML web page.

How to do it...

The regular expression pattern to match an e-mail address is as follows:

```
[A-Za-z0-9._]+@[A-Za-z0-9.]+\.[a-zA-Z]{2,4}
```

For example:

```
$ cat url_email.txt
this is a line of text contains,<email> #slynux@slynux.com. </email>
and email address, blog "http://www.google.com", test@yahoo.com
dfdfdfdddfdf;cool.hacks@gmail.com<br />
<a href="http://code.google.com"><h1>Heading</h1>
```

As we are using extended regular expressions (`+`, for instance), we should use `egrep`.

```
$ egrep -o '[A-Za-z0-9._]+@[A-Za-z0-9.]+\.[a-zA-Z]{2,4}'  url_email.txt
slynux@slynux.com
test@yahoo.com
cool.hacks@gmail.com
```

The `egrep` regex pattern for an HTTP URL is as follows:

```
http://[a-zA-Z0-9\-\.]+\.[a-zA-Z]{2,4}
```

For example:

```
$ egrep -o "http://[a-zA-Z0-9.]+\.[a-zA-Z]{2,3}" url_email.txt
http://www.google.com
http://code.google.com
```