- To start and end underlining use this:

```
tput smul
tput rmul
```

- To delete from the cursor to the end of the line use the following command:

```
tputed
```

- While typing a password, we should not display the characters typed. In the following example, we will see how to do it using `stty`:

```
#!/bin/sh
#Filename: password.sh
echo -e "Enter password: "
stty -echo
read password
stty echo
echo
echo Password read.
```

The `-echo` option in the preceding command disables the output to the terminal, whereas `echo` enables output.

Getting and setting dates and delays

Many applications require printing dates in different formats, setting date and time, and performing manipulations based on date and time. Delays are commonly used to provide a wait time (such as 1 second) during the program execution. Scripting contexts, such as monitoring a task every 5 seconds, demands the understanding of writing delays in a program. This recipe will show you how to work with dates and time delays.

Getting ready

Dates can be printed in variety of formats. We can also set dates from the command line. In Unix-like systems, dates are stored as an integer, which denotes the number of seconds since 1970-01-01 00:00:00 UTC. This is called **epoch** or **Unix time**. Let us see how to read dates and set them.

How to do it...

It is possible to read the dates in different formats and also to set the date. This can be accomplished with these steps:

1. You can read the date as follows:

   ```
   $ date
   Thu May 20 23:09:04 IST 2010
   ```

2. The epoch time can be printed as follows:

   ```
   $ date +%s
   1290047248
   ```

   We can find out epoch from a given formatted date string. You can use dates in multiple date formats as input. Usually, you don't need to bother about the date string format that you use if you are collecting the date from a system log or any standard application generated output. Convert the date string into epoch as follows:

   ```
   $ date --date "Thu Nov 18 08:07:21 IST 2010" +%s
   1290047841
   ```

   The `--date` option is used to provide a date string as input. However, we can use any date formatting options to print the output. Feeding the input date from a string can be used to find out the weekday, given the date.

   For example:

   ```
   $ date --date "Jan 20 2001" +%A
   Saturday
   ```

   The date format strings are listed in the table mentioned in the *How it works...* section:

3. Use a combination of format strings prefixed with `+` as an argument for the `date` command to print the date in the format of your choice. For example:

   ```
   $ date "+%d %B %Y"
   20 May 2010
   ```

4. We can set the date and time as follows:

   ```
   # date -s "Formatted date string"
   ```

   For example:

   ```
   # date -s "21 June 2009 11:01:22"
   ```