How to do it...

There are various operations you can perform on aliases, these are as follows:

1. An alias can be created as follows:

   ```
   $ alias new_command='command sequence'
   ```

   Giving a shortcut to the install command, `apt-get install`, can be done as follows:

   ```
   $ alias install='sudo apt-get install'
   ```

   Therefore, we can use `install pidgin` instead of `sudo apt-get install pidgin`.

2. The `alias` command is temporary; aliasing exists until we close the current terminal only. To keep these shortcuts permanent, add this statement to the `~/.bashrc` file. Commands in `~/.bashrc` are always executed when a new shell process is spawned:

   ```
   $ echo 'alias cmd="command seq"' >> ~/.bashrc
   ```

3. To remove an alias, remove its entry from `~/.bashrc` (if any) or use the `unalias` command. Alternatively, `alias example=` should unset the alias named `example`.

4. As an example, we can create an alias for `rm` so that it will delete the original and keep a copy in a backup directory:

   ```
   alias rm='cp $@ ~/backup && rm $@'
   ```

When you create an alias, if the item being aliased already exists, it will be replaced by this newly aliased command for that user.

There's more...

There are situations when aliasing can also be a security breach. See how to identify them.

Escaping aliases

The `alias` command can be used to alias any important command, and you may not always want to run the command using the alias. We can ignore any aliases currently defined by escaping the command we want to run. For example:

```
$ \command
```

The \ character escapes the command, running it without any aliased changes. While running privileged commands on an untrusted environment, it is always good security practice to ignore aliases by prefixing the command with \. The attacker might have aliased the privileged command with his/her own custom command to steal the critical information that is provided by the user to the command.

Grabbing information about the terminal

While writing command-line shell scripts, we will often need to heavily manipulate information about the current terminal, such as the number of columns, rows, cursor positions, masked password fields, and so on. This recipe helps in collecting and manipulating terminal settings.

Getting ready

`tput` and `stty` are utilities that can be used for terminal manipulations. Let us see how to use them to perform different tasks.

How to do it...

There are specific information you can gather about the terminal as shown in the following list:

- Get the number of columns and rows in a terminal by using the following commands:

  ```
  tput cols
  tput lines
  ```

- To print the current terminal name, use the following command:

  ```
  tput longname
  ```

- To move the cursor to a 100,100 position, you can enter:

  ```
  tput cup 100 100
  ```

- Set the background color for the terminal using the following command:

  ```
  tputsetb n
  ```

  `n` can be a value in the range of 0 to 7.

- Set the foreground color for text by using the following command:

  ```
  tputsetf n
  ```

  `n` can be a value in the range of 0 to 7.

- To make text bold use this:

  ```
  tput bold
  ```