- Separate index-value assignments:

```
$ ass_array[index1]=val1
$ ass_array'index2]=val2
```

For example, consider the assignment of prices for fruits, using an associative array:

```
$ declare -A fruits_value
$ fruits_value=([apple]='100 dollars' [orange]='150 dollars')
```

Display the contents of an array:

```
$ echo "Apple costs ${fruits_value[apple]}"
Apple costs 100 dollars
```

Listing of array indexes

Arrays have indexes for indexing each of the elements. Ordinary and associative arrays differ in terms of index type.

Obtain the list of indexes in an array.

```
$ echo ${!array_var[*]}
```

Alternatively, we can also use the following command:

```
$ echo ${!array_var[@]}
```

In the previous `fruits_value` array example, consider the following command:

```
$ echo ${!fruits_value[*]}
orange apple
```

This will work for ordinary arrays too.

Visiting aliases

An **alias** is a shortcut to replace typing a long-command sequence. In this recipe, we will see how to create aliases using the `alias` command.

How to do it...

These are the operations you can perform on aliases:

1. Create an alias:

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

   This example creates a shortcut for the `apt-get install` command:

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

   Once the alias is defined, we can type `install` instead of `sudo apt-get install`.

2. The `alias` command is temporary: aliases exist until we close the current terminal. To make an alias available to all shells, add this statement to the `~/.bashrc` file. Commands in `~/.bashrc` are always executed when a new interactive 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. This example creates an alias for `rm` that 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...

When running as a privileged user, aliases can be a security breach. To avoid compromising your system, you should escape commands.

Escaping aliases

Given how easy it is to create an alias to masquerade as a native command, you should not run aliased commands as a privileged user. We can ignore any aliases currently defined, by escaping the command we want to run. Consider this example:

```
$ \command
```

The `\` character escapes the command, running it without any aliased changes. When running privileged commands on an untrusted environment, it is always a 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 critical information that is provided by the user to the command.

Listing aliases

The `alias` command lists the currently defined aliases:

```
$ aliasalias lc='ls –color=auto'
alias ll='ls –l'
alias vi='vim'
```

Grabbing information about the terminal

While writing command-line shell scripts, we often need to 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

The `tput` and `stty` commands are utilities used for terminal manipulations.