Getting ready

The file command can be used to find out the type of the file by looking at the contents of the file. In Unix/Linux systems, file types are not determined based on the extension of the file (like the Microsoft Windows platform does). This recipe aims at collecting file type statistics of a number of files. For storing the count of files of the same type, we can use an associative array and the `file` command can be used to fetch the file type details from each of the files.

How to do it...

1. To print the type of a file use the following command:

```
$ file filename


$ file /etc/passwd
/etc/passwd: ASCII text
```

2. Print the file type only by excluding the filename as follows:

```
$ file -b filename
ASCII text
```

3. The script for file statistics is as follows:

```
#!/bin/bash
# Filename: filestat.sh

if [ $# -ne 1 ];
then
  echo "Usage is $0 basepath";
  exit
fi
path=$1

declare -A statarray;

while read line;
do
  ftype=`file -b "$line" | cut -d, -f1`
  let statarray["$ftype"]++;

done < <(find $path -type f -print)

echo ============ File types and counts =============
for ftype in "${!statarray[@]}";
do
  echo $ftype :  ${statarray["$ftype"]}
done
```

4. The usage is as follows:

   ```
   $ ./filestat.sh /home/slynux/temp
   ```

   A sample output is shown as follows:

   ```
   $ ./filetype.sh /home/slynux/programs
   ============ File types and counts =============
   Vim swap file : 1
   ELF 32-bit LSB executable : 6
   ASCII text : 2
   ASCII C program text : 10
   ```

How it works...

Here, an associative array named `statarray` is declared so that it can take the file type as file indices and store the count of each file type in the array. `let` is used to increment the count each time a file type is encountered. The `find` command is used to get the list of file paths recursively. A `while` loop is used to iterate line by line through the `find` command's output. The input line ``ftype=`file -b "$line"`` in the previous script is used to find out the file type using the `file` command. The `-b` option specifies the file command to print only the file type (without the filename in the output). The file type output consists of more details, such as image encoding used and resolution (in the case of an image file). But, we are not interested in all of the details; we need only the basic information. Details are comma-separated, as in the following example:

```
$ file a.out -b
ELF 32-bit LSB executable, Intel 80386, version 1 (SYSV), dynamically
linked (uses shared libs), for GNU/Linux 2.6.15, not stripped
```

We need to extract only `ELF 32-bit LSB executable` from the previous details. Hence, we use `cut -d, -f1`, which specifies to use `,` as the delimiter and print only the first field.

`done < <(find $path -type f -print);` is an important bit of code. The logic is as follows:

```
while read line;
do something
done < filename
```

Instead of the filename we used the output of `find`.

`<(find $path -type f -print)` is equivalent to a filename. But it substitutes filename with a subprocess output. Note that the first `<` is for input redirection and the second `<` is for converting the subprocess output to a filename. Also, there is a space between these two so that the shell won't interpret it as the `<<` operator.