```
  sleep $UNIT_TIME
done

echo
echo CPU eaters :

cat /tmp/cpu_usage.$$ | \
awk '
{ process[$1]+=$2; }
END{
  for(i in process)
  {
    printf("%-20s %s\n",i, process[i]) ;
  }

   }' | sort -nrk 2 | head

rm /tmp/cpu_usage.$$
#Remove the temporary log file
```

A sample output is as follows:

```
$ ./pcpu_usage.sh
Watching CPU usage...
CPU eaters :
Xorg        20
firefox-bin   15
bash        3
evince      2
pulseaudio    1.0
pcpu.sh         0.3
wpa_supplicant  0
wnck-applet     0
watchdog/0      0
usb-storage     0
```

How it works...

In the preceding script, the major input source is `ps -eocomm,pcpu`. `comm` stands for command name and `pcpu` stands for the CPU usage in percent. It will output all the process names and the CPU usage in percent. For each process there exists a line in the output. Since we need to monitor the CPU usage for one hour, we repeatedly take usage statistics using `ps -eocomm,pcpu | tail -n +2` and append to a file `/tmp/cpu_usage.$$` running inside a `for` loop with 60 seconds wait in each iteration. This wait is provided by `sleep 60`. It will execute `ps` once in each minute. `tail -n +2` is used to strip off the header and `COMMAND %CPU` in the `ps` output.

`$$ in cpu_usage.$$` signifies that it is the process ID of the current script. Suppose PID is `1345`; during execution it will be replaced as `/tmp/cpu_usage.1345`. We place this file in `/tmp` since it is a temporary file.

The statistics file will be ready after one hour and will contain 60 sets of entries, each set containing entries corresponding to the process status for each minute. Then `awk` is used to sum the total CPU usage for each process. An associative array process is used for the summation of CPU usages. It uses the process name as array index. Finally, it sorts the result with a numeric reverse sort according to the total CPU usage and pass through `head` to obtain the top 10 usage entries.

See also

- The *Using awk for advanced text processing* recipe of *Chapter 4*, *Texting and Driving*, explains the `awk` command
- The *Using head and tail for printing the last or first 10 lines* recipe of *Chapter 3*, *File In, File Out*, explains the `tail` command

Monitoring command outputs with watch

We might need to continuously watch the output of a command for a period of time in equal intervals. For example, while copying a large file, we might need to watch the growth of the file size. In order to do that, we can use the `watch` command to execute the `du` command and output repeatedly. This recipe explains how to do that.

How to do it...

The `watch` command can be used to monitor the output of a command on the terminal at regular intervals. The syntax of the `watch` command is as follows:

```
$ watch COMMAND
```