An executable binary of the `time` command is available at `/usr/bin/time`, as well as a shell built-in named `time` exists. When we run time, it calls the shell built-in by default. The shell built-in `time` has limited options. Hence, we should use an absolute path for the executable (`/usr/bin/time`) for performing additional functionalities.

2. We can write these time statistics to a file using the -o filename option as follows:

```
$ /usr/bin/time -o output.txt COMMAND
```

The filename should always appear after the `-o` flag.

In order to append the time statistics to a file without overwriting, use the `-a` flag along with the `-o` option as follows:

```
$ /usr/bin/time -a -o output.txt COMMAND
```

3. We can also format the time outputs using format strings with `-f` option. A format string consists of parameters corresponding to specific options prefixed with `%`. Format strings for real time, user time, and sys time are as follows:

   - Real time: `%e`
   - User: `%U`
   - sys: `%S`

By combining parameter strings, we can create a formatted output as follows:

```
$ /usr/bin/time -f "FORMAT STRING" COMMAND
```

For example:

```
$ /usr/bin/time -f "Time: %U" -a -o timing.log uname
Linux
```

Here `%U` is the parameter for user time.

When a formatted output is produced, the formatted output of the command is written to the standard output and the output of the `COMMAND`, which is timed, is written to standard error. We can redirect the formatted output using a redirection operator (`>`) and redirect the time information output using the (`2>`) error redirection operator.

For example:

```
$ /usr/bin/time -f "Time: %U" uname> command_output.txt 2>time.log
$ cat time.log
Time: 0.00
$ cat command_output.txt
Linux
```

4. To show the page size, use the `%Z` parameters as follows:

```
$ /usr/bin/time -f "Page size: %Z bytes" ls> /dev/null
Page size: 4096 bytes
```

   Here the output of the timed command is not required and hence, the standard output is directed to the `/dev/null` device in order to prevent it from writing to the terminal.

More format string parameters are available. Try `man time` for more details.

How it works...

The three different times can be defined as follows:

- **Real** is wall clock time—the time from start to finish of the call. This is all elapsed time including time slices used by other processes and the time that the process spends when blocked (for example, if it is waiting for I/O to complete).
- **User** is the amount of CPU time spent in user-mode code (outside the kernel) within the process. This is only the actual CPU time used in executing the process. Other processes, and the time that the process spends when blocked do not count towards this figure.
- **Sys** is the amount of CPU time spent in the kernel within the process. This means executing the CPU time spent in system calls within the kernel, as opposed to the library code, which is still running in the user space. Like user time, this is only the CPU time used by the process. Refer to the following table for a brief description of kernel mode (also known as supervisor mode) and the system call mechanism.

Many details regarding a process can be collected using the `time` command. The important details include, exit status, number of signals received, number of context switches made, and so on. Each parameter can be displayed by using a suitable format string.

The following table shows some of the interesting parameters that can be used:

Parameter	Description
`%C`	Name and command-line arguments of the command being timed.
`%D`	Average size of the process's unshared data area, in kilobytes.
`%E`	Elapsed real (wall clock) time used by the process in [hours:]minutes:seconds.
`%x`	Exit status of the command.
`%k`	Number of signals delivered to the process.
`%W`	Number of times the process was swapped out of the main memory.
`%Z`	System's page size in bytes. This is a per-system constant, but varies between systems.