For example, to list the top 10 CPU consuming processes, use:

```
$ ps -eo comm,pcpu --sort -pcpu | head
COMMAND          %CPU
Xorg              0.1
hald-addon-stor   0.0
ata/0             0.0
scsi_eh_0         0.0
gnome-settings-   0.0
init              0.0
hald              0.0
pulseaudio        0.0
gdm-simple-gree   0.0
```

Here, processes are sorted in the descending order by percentage of CPU usage, and `head` is applied to extract the top 10 processes.

We can use `grep` to extract entries in the `ps` output related to a given process name or another parameter. In order to find out entries about running Bash processes, use:

```
$ ps -eo comm,pid,pcpu,pmem | grep bash
bash              1255  0.0  0.3
bash              1680  5.5  0.3
```

Finding the process ID when given command names

Suppose several instances of a command are being executed, we may need to identify the PID of the processes. This information can be found by using the `ps` or the `pgrep` command. We can use `ps` as follows:

```
$ ps -C COMMAND_NAME
```

Or

```
$ ps -C COMMAND_NAME -o pid=
```

The `-o` user-defined format specifier was described in the earlier part of the recipe. But here, you can see `=` appended with `pid`. This is to remove the header PID in the output of `ps`. In order to remove headers for each column, append `=` to the parameter. For example:

```
$ ps -C bash -o pid=
 1255
 1680
```

This command lists the process IDs of Bash processes.

Alternately, there is a handy command called `pgrep`. You should use `pgrep` to get a quick list of process IDs for a particular command. For example:

```
$ pgrep COMMAND
$ pgrep bash
1255
1680
```

> `pgrep` requires only a portion of the command name as its input argument to extract a Bash command, for example, `pgrep ash` or `pgrep bas` will also work. But `ps` requires you to type the exact command.

`pgrep` accepts many more output-filtering options. In order to specify a delimiter character for output rather than using a newline as the delimiter, use:

```
$ pgrep COMMAND -d DELIMITER_STRING
$ pgrep bash -d ":"
1255:1680
```

Specify a list of owners of the user for the matching processes as follows:

```
$ pgrep -u root,slynux COMMAND
```

In this command, `root` and `slynux` are users. Return the count of matching processes as follows:

```
$ pgrep -c COMMAND
```

Filters with ps for real user or ID, effective user or ID

With `ps`, it is possible to group processes based on the real and effective username or ID specified. Specified arguments can be used to filter the `ps` output by checking whether each entry belongs to a specific, effective user, or real user from the list of arguments and shows only the entries matching them. This can be done as follows:

- Specify an effective users' list by using `-u EUSER1`, `EUSER2`, and so on
- Specify a real users' list by using `-U RUSER1`, `RUSER2`, and so on

For example:

```
$ ps -u root -U root -o user,pcpu
```

This command will show all processes running with `root` as the effective user ID and real user ID, and will also show the user and percentage CPU usage columns.