

Process Metrics and Process Control

Load Averages:

1. The load average is the average of the load number for a given period of time.
2. It takes into account processes that are:
 - Actively running on a CPU.
 - Considered runnable, but waiting for a CPU to become available.
 - Sleeping: i.e. waiting for some kind of resource (typically, I/O) to become available.
3. **NOTE:** Linux differs from other UNIX-like operating systems in that it includes the sleeping processes. Furthermore, it only includes so-called uninterruptible sleepers, those which cannot be awakened easily.
4. The load average can be viewed by running `w`, `top` or `uptime`.

Interpreting Load Averages:

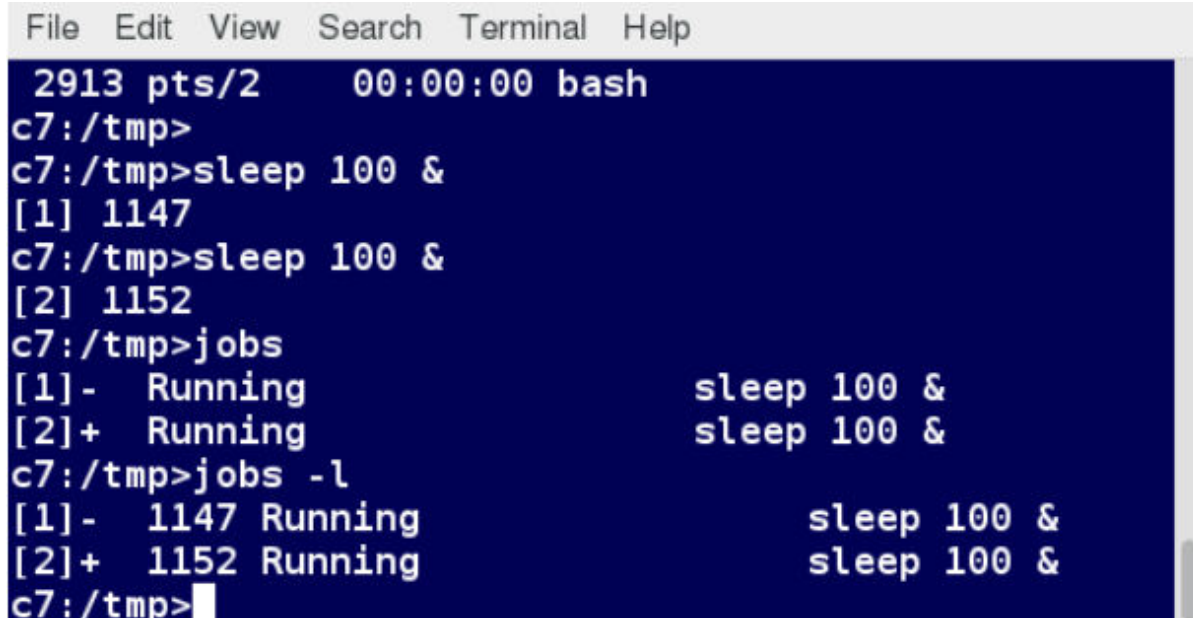
1. The load average is displayed using three numbers (0.45, 0.17, and 0.12) in the below screenshot.
2. Assuming our system is a single-CPU system, the three load average numbers are interpreted as follows:
 - 0.45: For the last minute the system has been 45% utilized on average.
 - 0.17: For the last 5 minutes utilization has been 17%.
 - 0.12: For the last 15 minutes utilization has been 12%.
3. If we saw a value of 1.00 in the second position, that would imply that the single-CPU system was 100% utilized, on average, over the past 5 minutes; this is good if we want to fully use a system.
4. A value over 1.00 for a single-CPU system implies that the system was over-utilized: there were more processes needing CPU than CPU was available.

Background and Foreground Processes:

1. Linux supports background and foreground job processing. Job in this context is just a command launched from a terminal window.
2. Foreground jobs run directly from the shell, and when one foreground job is running, other jobs need to wait for shell access (at least in that terminal window if using the GUI) until it is completed.
3. This is fine when jobs complete quickly. But this can have an adverse effect if the current job is going to take a long time (even several hours) to complete.
4. In such cases, you can run the job in the background and free the shell for other tasks.
5. The background job will be executed at lower priority, which, in turn, will allow smooth execution of the interactive tasks, and you can type other commands in the terminal window while the background job is running.
6. By default, all jobs are executed in the foreground. You can put a job in the background by suffixing **&** to the command, for example: **updatedb &**.
7. You can either use **CTRL-Z** to suspend a foreground job or **CTRL-C** to terminate a foreground job and can always use the **bg** and **fg** commands to run a process in the background and foreground, respectively.

Managing Jobs:

1. The jobs utility displays all jobs running in background. The display shows the job ID, state, and command name, as shown in the below screenshot.
2. **jobs -l** provides the same information as jobs, and adds the **PID** of the background jobs.
3. The background jobs are connected to the terminal window, so, if you log off, the jobs utility will not show the ones started from that window.

A screenshot of a terminal window with a dark blue background and white text. The window has a menu bar at the top with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. The terminal shows a user at a prompt 'c7:/tmp>' running several commands. First, 'sleep 100 &' is run, returning '[1] 1147'. Then, another 'sleep 100 &' is run, returning '[2] 1152'. Next, the 'jobs' command is run, showing '[1]- Running sleep 100 &' and '[2]+ Running sleep 100 &'. Finally, 'jobs -l' is run, showing '[1]- 1147 Running sleep 100 &' and '[2]+ 1152 Running sleep 100 &'. The prompt 'c7:/tmp>' is visible at the bottom with a cursor.

```
File Edit View Search Terminal Help
2913 pts/2      00:00:00 bash
c7:/tmp>
c7:/tmp>sleep 100 &
[1] 1147
c7:/tmp>sleep 100 &
[2] 1152
c7:/tmp>jobs
[1]-  Running          sleep 100 &
[2]+  Running          sleep 100 &
c7:/tmp>jobs -l
[1]-  1147 Running      sleep 100 &
[2]+  1152 Running      sleep 100 &
c7:/tmp>
```