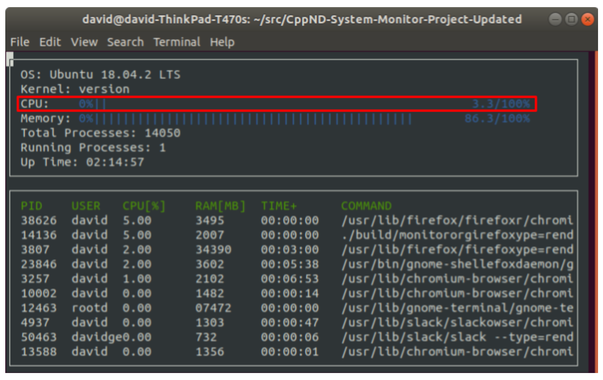
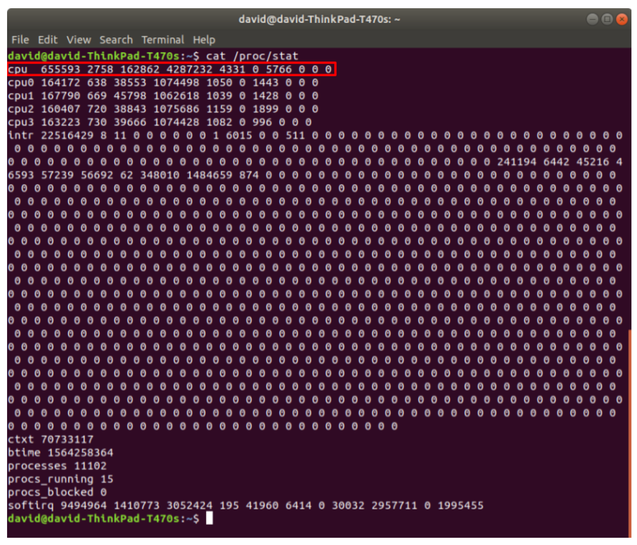
Processor Data

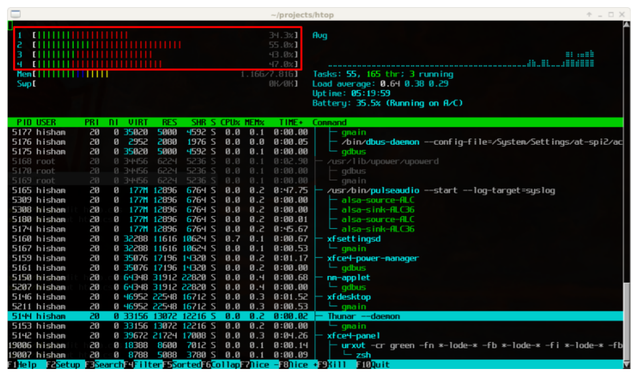


Linux stores processor utilization data within the `/proc/stat` file.



This data is more complex than most of the other data necessary to complete this project.

For example, `/proc/stat` contains aggregate processor information (on the "cpu" line) and individual processor information (on the "cpu0", "cpu1", etc. lines). Indeed, **htop** displays utilization information for each individual processor.



For this project, however, you only need to display aggregate CPU information, which you can find on the "cpu" line of `/proc/stat`.

If you would like to add individual processor information to your system monitor project, go for it!

Data

`/proc/stat` contains 10 integer values for each processor. The Linux source code **documents each of these numbers**:

> The very first "cpu" line aggregates the numbers in all of the other "cpuN" lines. These numbers identify the amount of time the CPU has spent performing different kinds of work. Time units are in USER_HZ (typically hundredths of a second). The meanings of the columns are as follows, from left to right:
>
> - user: normal processes executing in user mode
> - nice: niced processes executing in user mode
> - system: processes executing in kernel mode
> - idle: twiddling thumbs
> - iowait: In a word, iowait stands for waiting for I/O to complete. But there are several problems:
>   1. Cpu will not wait for I/O to complete, iowait is the time that a task is waiting for I/O to complete. When cpu goes into idle state for outstanding task io, another task will be scheduled on this CPU.
>   2. In a multi-core CPU, the task waiting for I/O to complete is not running on any CPU, so the iowait of each CPU is difficult to calculate.
>   3. The value of iowait field in /proc/stat will decrease in certain conditions. So, the iowait is not reliable by reading from /proc/stat.
> - irq: servicing interrupts
> - softirq: servicing softirqs
> - steal: involuntary wait
> - guest: running a normal guest
> - guest_nice: running a niced guest

Even once you know what each of these numbers represents, it's still a challenge to determine exactly how to use these figures to calculate processor utilization. **This guide** and **this StackOverflow post** are helpful.

Measurement Interval

Once you've parsed `/proc/stat` and calculated the processor utilization, you've got what you need for this project. Congratulations!

However, when you run your system monitor, you might notice that the process utilization seems very stable. Too stable.

That's because the processor data in `/proc/stat` is measured since boot. If the system has been up for a long time, a temporary interval of even extreme system utilization is unlikely to change the long-term average statistics very much. This means that the processor could be red-lining *right now* but the system monitor might still show a relatively underutilized processor, if the processor has spent most of the time since boot in an idle state.

You might want to update the system monitor to report the current utilization of the processor, rather than the long-term average utilization since boot. You would need to measure the difference in system utilization between two points in time relatively close to the present. A formula like:

Δ active time units / Δ total time units

Consider this a bonus challenge that is not required to pass the project.