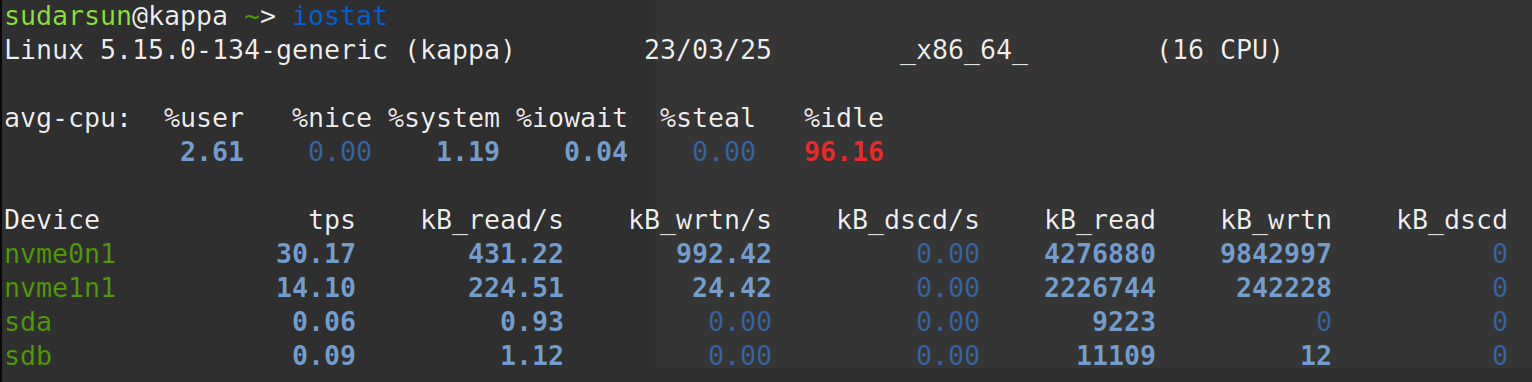
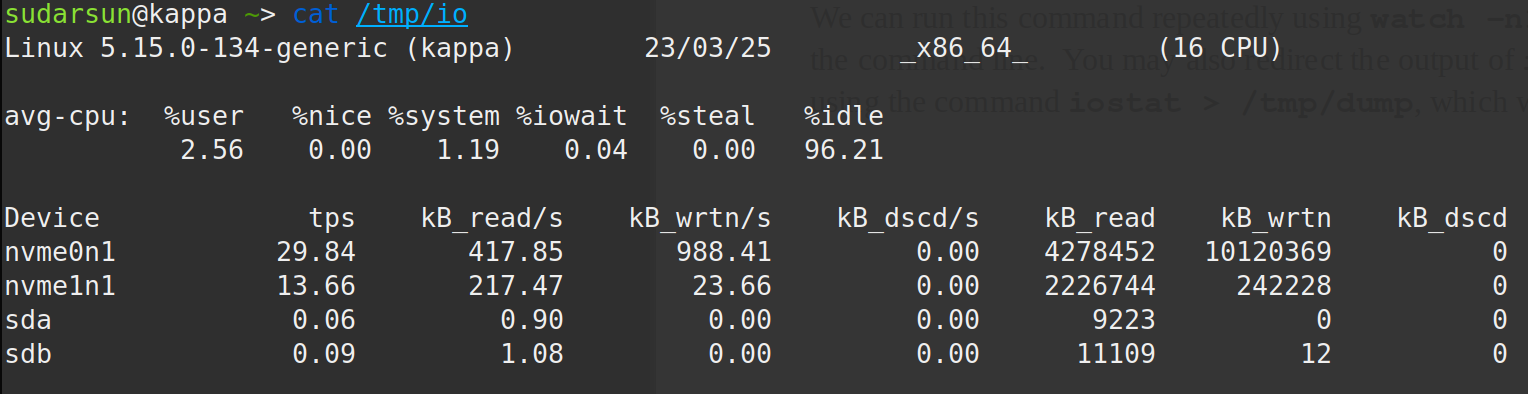
## DA5402: Assignment 6

Let’s now learn to build an AI application with instrumentation using Prometheus. We learned how to use exporters to monitor standard application and about building an exporter for our application via Promotheus instrumentation library. Let’s try creating an exporter for exporting the system (node) level metrics without using **node\_exporter** or **windows\_exporter**.

### Task 1 [20 points]

In the Linux environment, we have a command name **iostat** which lists the disk level IO statistics like the snapshot below.

  
We can run this command repeatedly using **watch -n1 iostat** to have a live monitoring from the command line. You may also redirect the output of **iostat** into a file for further processing using the command **iostat > /tmp/io**, which will appear like below.

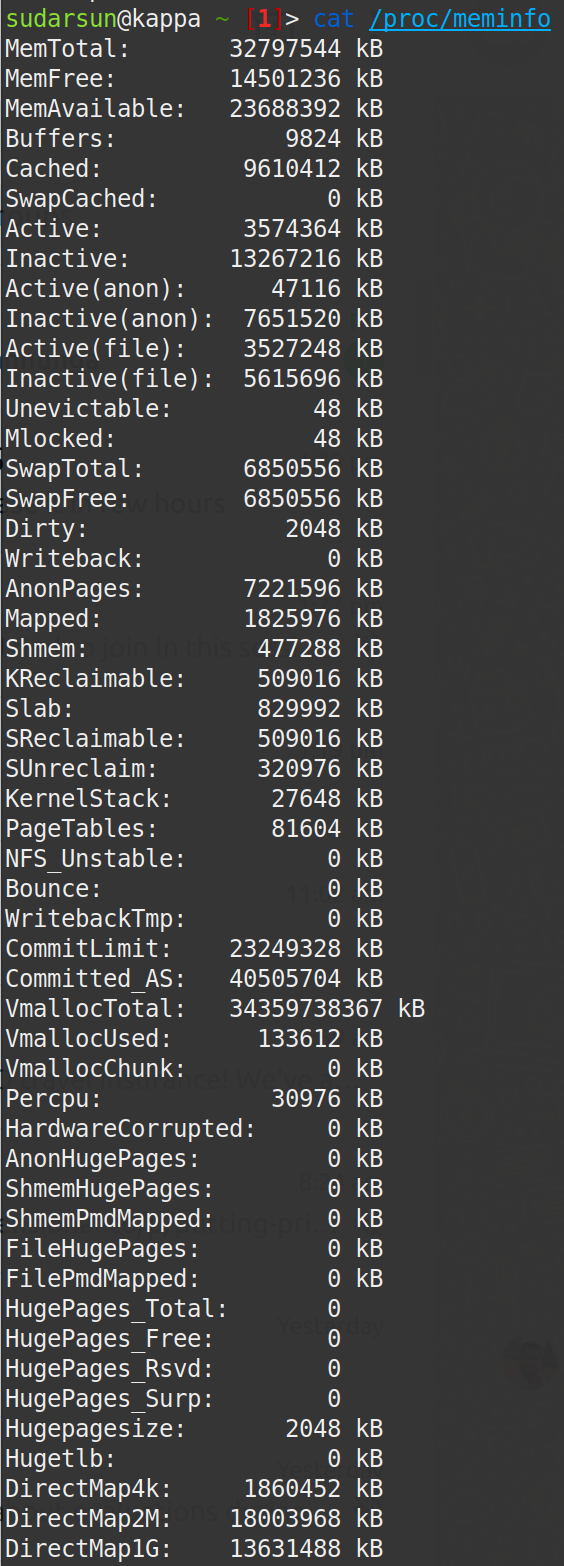


You can process the text dump to collect the statistics. The objective here is to convert the collected statistics into suitable Prometheus metric types with appropriate names. And you repeat the data collection every second (using **crontab** or via scripting), so Prometheus service can pick the metrics at the **scrape\_interval >= 1s**. The metrics to be exported are:

* io\_read\_rate{device=``xxx”}
* io\_write\_rate{device=``xxx”}
* io\_tps{device=``xxx”}
* io\_read\_bytes{device=``xxx”}
* io\_write\_bytes{device=``xxx”}
* cpu\_avg\_percent{mode=``xxx”} xxx \in {user, nice, system, iowait, idle}

### Task 2 [20 points]

Likewise, we can read the current memory information from **/proc/meminfo**, which appears like below:



The file updates live, which can be observed using **watch -n1 cat /proc/meminfo** command. Similar to Task 1, you should repeatedly read from **/proc/meminfo** every second to monitor the live memory statistics, followed by exporting them as Prometheus metrics. Every line item from the meminfo listing should be exported as a metric. Every metric from meminfo should have **meminfo\_** as the prefix. For instance the line item **MemFree** should become **meminfo\_free\_memory** as the metric.

Note the, both tasks should be implemented as individual functions in a single Python script, which would expose the metrics at port 18000. So, <http://localhost:18000/metrics> should report all the exported metrics from both functions. As usual, use of logging mechanism is mandatory. Comments around the source code block is crucial.

### Task 3 [10 points]

Setup Prometheus server and configure it to scrape from your instrumented application by setting the scrape interval to 2 seconds. Ensure that the metrics your exposed from the app are queryable from Prometheus UI console.