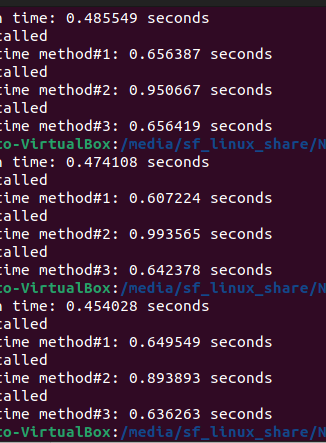
**Report**



Execution times for each method in c language

**Description:**

After comparing, we observe that method#2, in which only one row is read from the file at a time and processed, has the most time-consuming. The reason is that we open the file to get one line of data and then process it. After processing one line, we open the file again to get the next row of data and process. By following this approach, we can conclude that we open the file as many times as the number of lines in the file (which in this case is a million). But, in method#1, where the file opens one time only, to get all the data from the file to a buffer and then process it, we retrieve all the data in one go. Likewise, in method#3, where the file opens two times only, to get 1st half and then 2nd half of the data and process it each time, we retrieve all the data in two trips.

Let's put this in simple example words, consider you have to transport 100 paper sheets from the ground floor of a building to its top floor. If you transport the paper sheets one by one, you are climbing the stairs 100 times to get all the sheets from source to destination, which is, of course, more time and energy-consuming. This is how we are working in method#2.

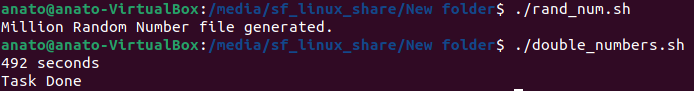
Compared to the above, if you transport the paper sheets 50 at a time, you take two trips to get all the sheets from source to destination, which is much shorter and more effective. This is how we are working in method#3.

Compared to the both in the above, if you transport the paper sheets 100 at a time, you take only one trip to get all the sheets from source to destination, which is the shortest and most effective. This is how we are working in method#1.

Therefore, after comparison, it is concluded that the method#1 is the most recommended method for analyzing and processing data while processing data from files

The results match my expectations as it is clear after giving proper thought that more trips will eventually mean more time and energy consumption.

­­­­­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_



**Comparison with Bash Script:**

After comparing, we observe that the c language code executed much faster that Bash.

Bash will be slower than C for the actual runtime.

However, the use case for bash isn’t execution speed - it’s ease of gluing together other system commands and components. And often, the total time for “recognize need, code solution, get answer” is a \*lot\* lower for Bash, because you can create a one-liner and an answer a lot faster.