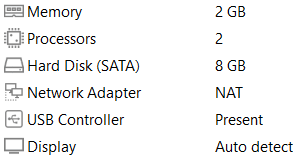
Отчёт по лабораторной работе номер 6.

Параметры из вывода команды **top**:

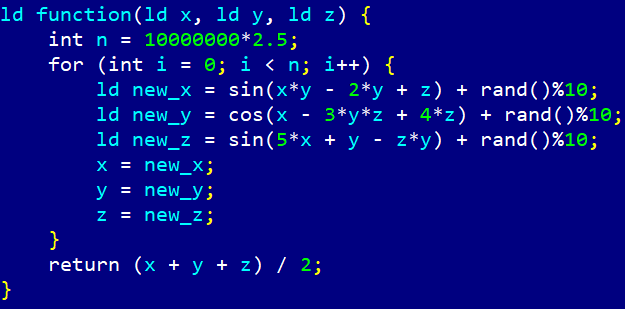


Параметры виртуальной машины:

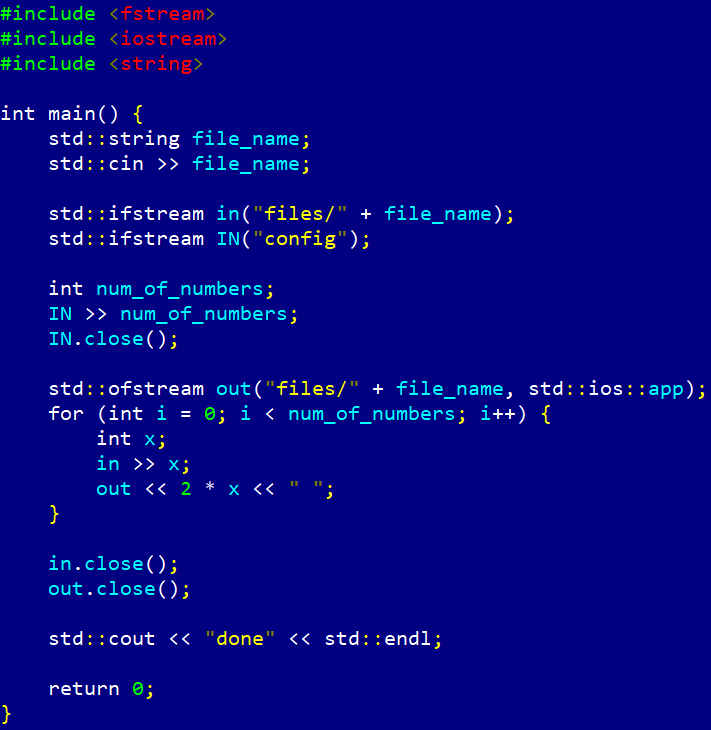


Алгоритмы:

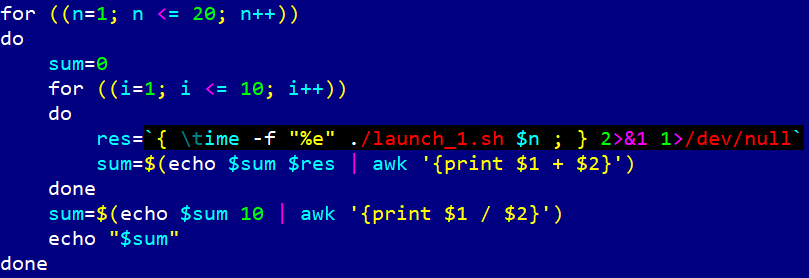
В качестве первого алгоритма я взял вычисление странной функции в R^3:



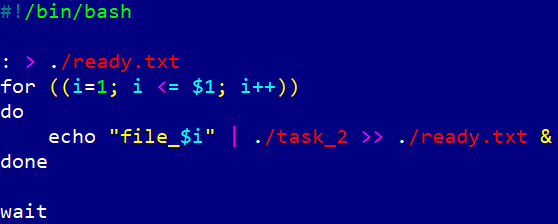
Второй алгоритм для работы с файлами:



Скрипт для подсчёта средних значений:

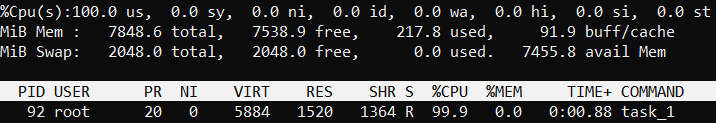


Пример запускающего скрипта:



1. Группа экспериментов номер 1
   1. Последовательное выполнение, 1 процессор

Мы видим линейную зависимость, т.к. у нас один процессор и послед. выполнение.



На этом скрине мы видим, что память на нуле, т.к. мы её вообще не используем, cpu загружен на 100%.

* 1. Последовательное выполнение, 2 процессора

Аналогично лин. зависимость, всё выполняется последовательно, поэтому общее время выполнения такое же.



Здесь мы видим, что приоритет отдаётся одному из процессоров (бывает, что и второму), оба не используюся, т.к. запуск послед.

Аналогично пункту 1.2 главная задача - это task\_1, память на нуле.

* 1. Параллельное выполнение, 1 процессор

Хоть выполнение и параллельное, процессор то 1, поэтому не видно никакого эффекта.



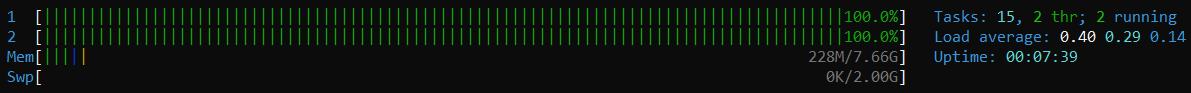
Единственный процессор загружен на макс.



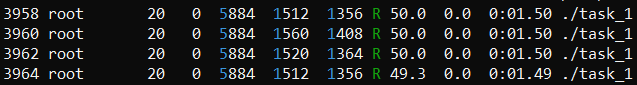
Ресурсы распр. справедливо.

* 1. Параллельное выполнение, 2 процессора

Этот случай интереснее. На 1 запуске мы потратили ~3 секунды, на 2 запуске у нас задачи работали параллельно на двух процессорах, поэтому время тоже ~3 секунды, даже чуть меньше. Далее мы наблюдаем линейный рост, но коэффициент стал в 2 раза меньше, т.к. у нас два проц.



Здесь, в отличие от п.1.2., оба процессора загружены полностью.



Распределение ресурсов справедливое.

1. Группа экспериментов номер 2

У меня при N=4E6 программа работает 2-3 секунды.

* 1. Последовательное выполнение, 1 процессор

Аналогично имеем лин. рост.

* 1. Последовательное выполнение, 2 процессора

Выполнение послед., процессора 2, но меняться ничего не должно, поэтому мы видим лин. завис. с таким же углом.

* 1. Параллельное выполнение, 1 процессор

Опять же ничего не меняется, всё та же верхняя граница, так же лин. зависимость.

* 1. Параллельное выполнение, 2 процессора

Стало 2 процессора, поэтому программа стала работать в 2 раза быстрее.

В последних 4х экспериментах cpu загружены не полностью, т.к. основная работа идёт с памятью.

Вывод: если задача не распр., то не важно сколько у нас процессоров.