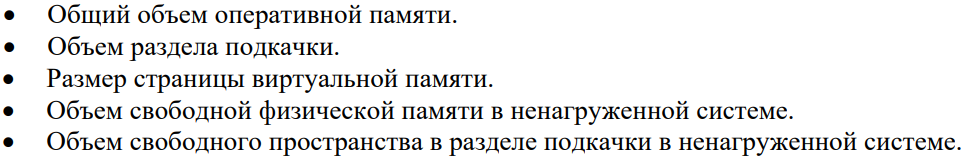
1. Текущая конфигурация операционной системы в аспекте управления памятью.

Запустим команду top:



Отсюда узнаем параметры из условия:



Значения: 1827 MiB, 820 MiB, 1432.5 MiB, 820 MiB.

Третий параметр узнаем такой командой: , значение - 4096.

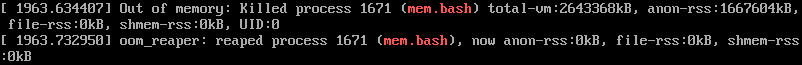
1. Эксперимент №1.

Последняя запись в файле report.log, т.е. размер массива: 30000000.

Последняя запись системного журнала:



Результат вывода команды :



Графики для одного процесса:

Выполним команду dmesg | grep "mem[2]\*.bash":

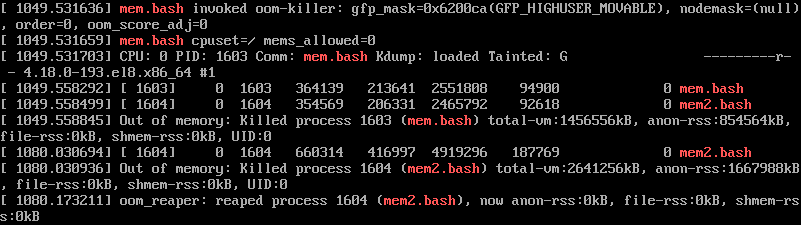


График для двух процессов:

Последнии записи в файлах report.log и report2.log соответственно: 30000000 и 15000000.

Вывод: В первом эксперименте процесс начал сначала заполнять физическую память, когда она закончилась, то он начал заполнять виртуальную память, когда закончилась и та, тогда процесс завершился аварийно.

Во втором эксперименте происходило тоже самое, только сначала убился один процесс и освободилась ровно половина вирт. и физ. памяти, потом уже убился и второй процесс. По числам 15 млн и 30 млн мы видим, что процессы заполняли память симметрично.

1. Эксперимент №2.

Запустим 10 процессов и действительно увидим, что они все завершились штатно. Запустим 30 процессов и увидим, что часть из них завершилась аварийно (10 завершилось штатно). Поменяем k на 30. Можно запустить скрипт с бинарным поиском, но я подобрал руками. У меня получилось, что при 2 млн ровно программа работала штатно, а при 2 млн и 10 тыс. программа падала аварийно. Это вполне логично. Изначально у меня падало при 30 млн., мы создали 10 процессов по 3 млн., задержка была 1 сек., процессы успевали завершаться и освобождать память для других. Теперь мы увеличили k в 3 раза, значит наше значение должно лежать между 1 млн. и 3 млн. Так и получилось. Процессы освобождали память и давали память другим.