1. Текущая конфигурация операционной системы в аспекте управления памятью.

Запустим команду top:

```
MiB Mem :   1827.0 total,   1432.5 free,    171.3 used,    223.1 buff/cache
MiB Swap:    820.0 total,    820.0 free,      0.0 used.   1501.7 avail Mem
```

Отсюда узнаем параметры из условия:

- Общий объем оперативной памяти.
- Объем раздела подкачки.
- Размер страницы виртуальной памяти.
- Объем свободной физической памяти в ненагруженной системе.
- Объем свободного пространства в разделе подкачки в ненагруженной системе.

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

Третий параметр узнаем такой командой: `getconf PAGE_SIZE`, значение - 4096.

2. Эксперимент №1.

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

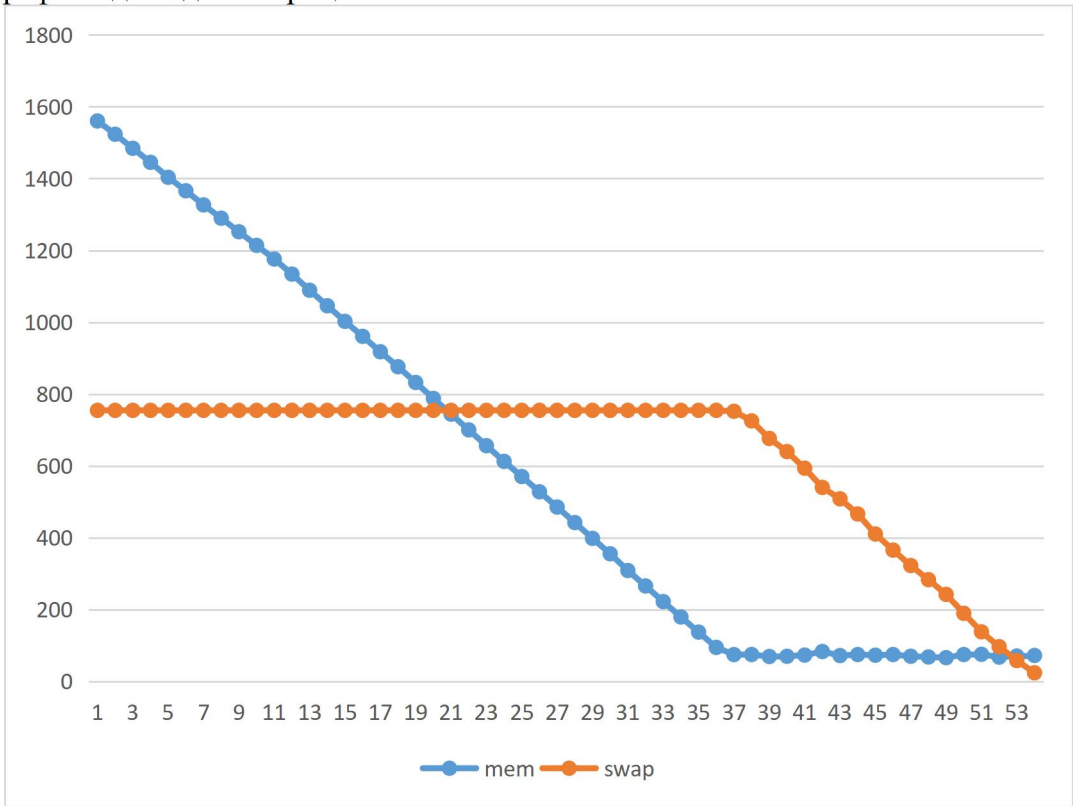
Последняя запись системного журнала:

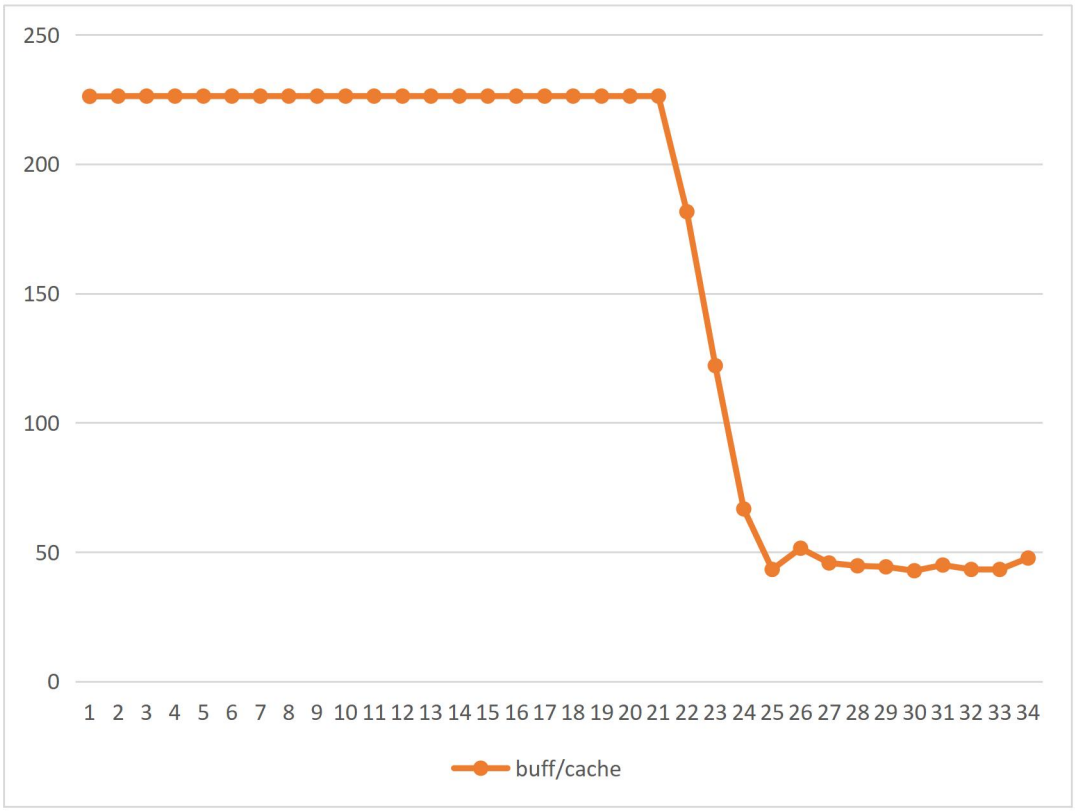
```
[ 1312.573576] Out of memory: Killed process 1635 (mem.bash) total-vm:2642312kB, anon-rss:1642080kB,
 file-rss:0kB, shmem-rss:0kB, UID:0
[ 1312.725597] oom_reaper: reaped process 1635 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss
:0kB
```

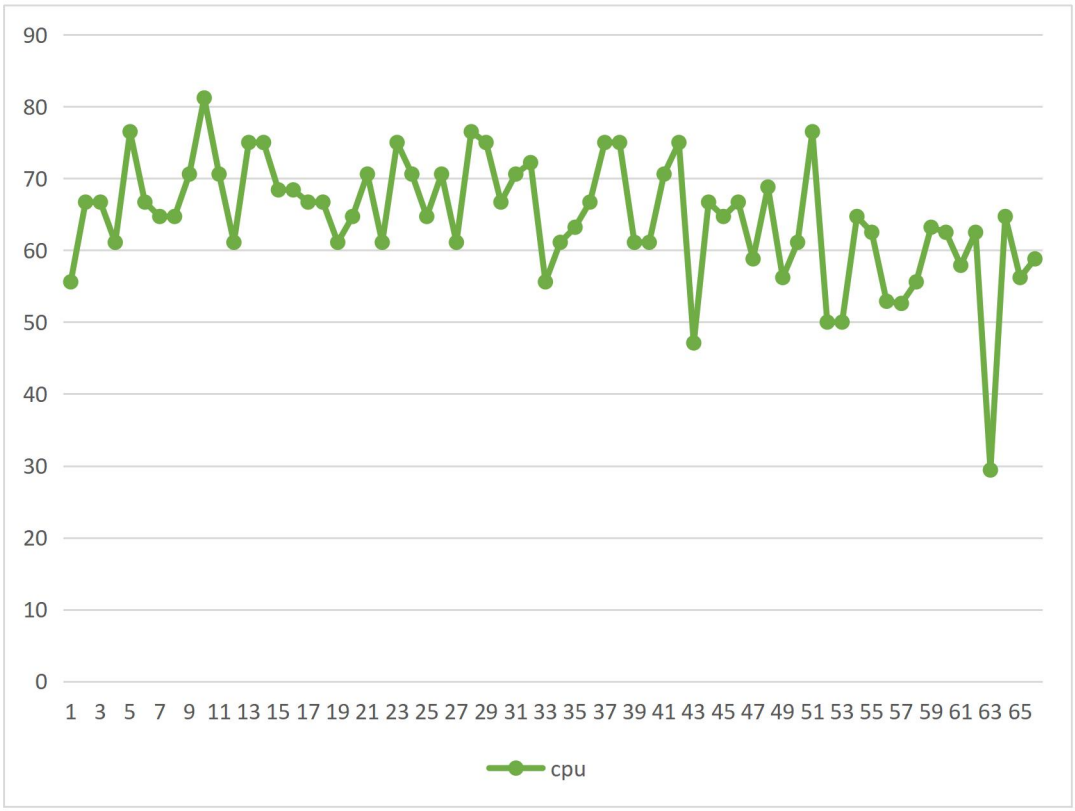
Результат вывода команды `dmesg | grep "mem.bash"`:

```
[ 1963.634407] Out of memory: Killed process 1671 (mem.bash) total-vm:2643368kB, anon-rss:1667604kB,
 file-rss:0kB, shmem-rss:0kB, UID:0
[ 1963.732950] oom_reaper: reaped process 1671 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss
:0kB
```

Графики для одного процесса:



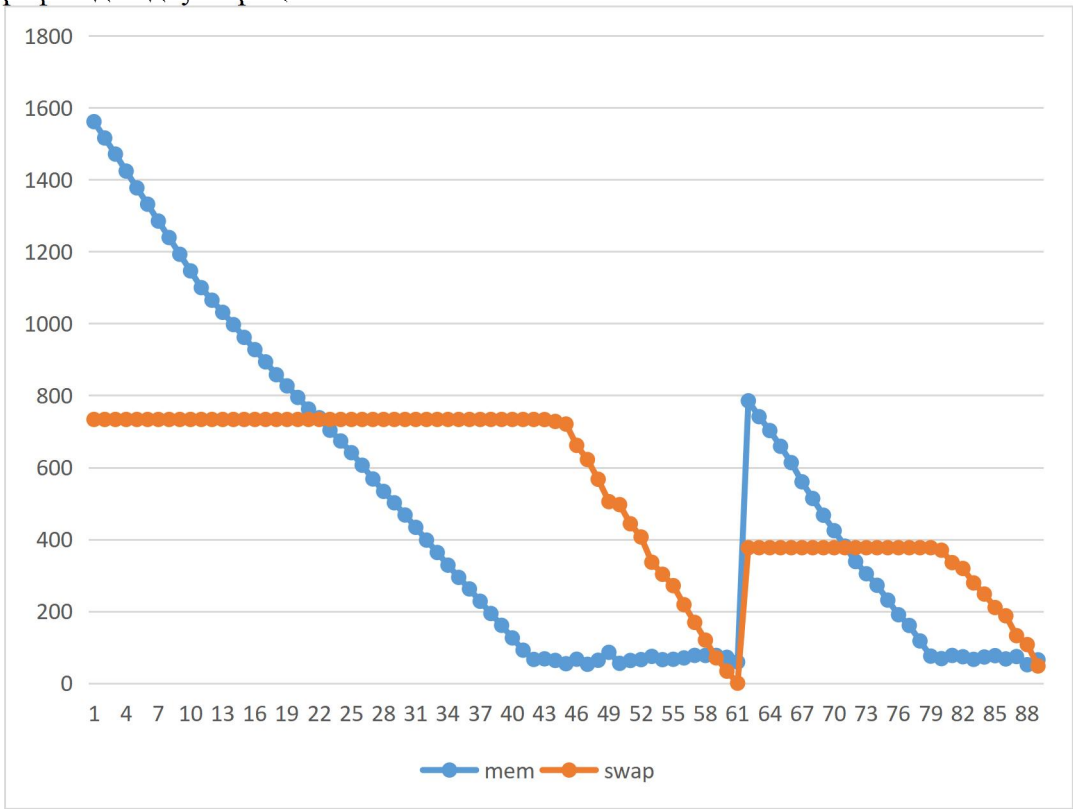

250
200
150
100
50
0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34
buff/cache


90
80
70
60
50
40
30
20
10
0
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63 65
cpu

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

```
[ 1049.531636] mem.bash invoked oom-killer: gfp_mask=0x6200ca(GFP_HIGHUSER_MOVABLE), nodemask=(null)
, order=0, oom_score_adj=0
[ 1049.531659] mem.bash cpuset=/ mems_allowed=0
[ 1049.531703] CPU: 0 PID: 1603 Comm: mem.bash Kdump: loaded Tainted: G                ---------r-
 - 4.18.0-193.el8.x86_64 #1
[ 1049.558292] [ 1603]     0  1603   364139   213641  2551808    94900             0 mem.bash
[ 1049.558499] [ 1604]     0  1604   354569   206331  2465792    92618             0 mem2.bash
[ 1049.558845] Out of memory: Killed process 1603 (mem.bash) total-vm:1456556kB, anon-rss:854564kB,
file-rss:0kB, shmem-rss:0kB, UID:0
[ 1080.030694] [ 1604]     0  1604   660314   416997  4919296   187769             0 mem2.bash
[ 1080.030936] Out of memory: Killed process 1604 (mem2.bash) total-vm:2641256kB, anon-rss:1667988kB
, file-rss:0kB, shmem-rss:0kB, UID:0
[ 1080.173211] oom_reaper: reaped process 1604 (mem2.bash), now anon-rss:0kB, file-rss:0kB, shmem-rs
s:0kB
```

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.

3. 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.