Отчет по лабораторной работе №5

Назаров Рустам M3232

12 декабря 2023г

Конфигурация системы:

Объем оперативной памяти: 1.8Gb

Объем раздела подкачки: 819Mb

Размер страницы виртуальной памяти: 4096b

Объем свободной физической памяти в ненагруженной системе: 1.6Gb

Объем свободного пространства в разделе подкачки в ненагруженной системе: 731Mb

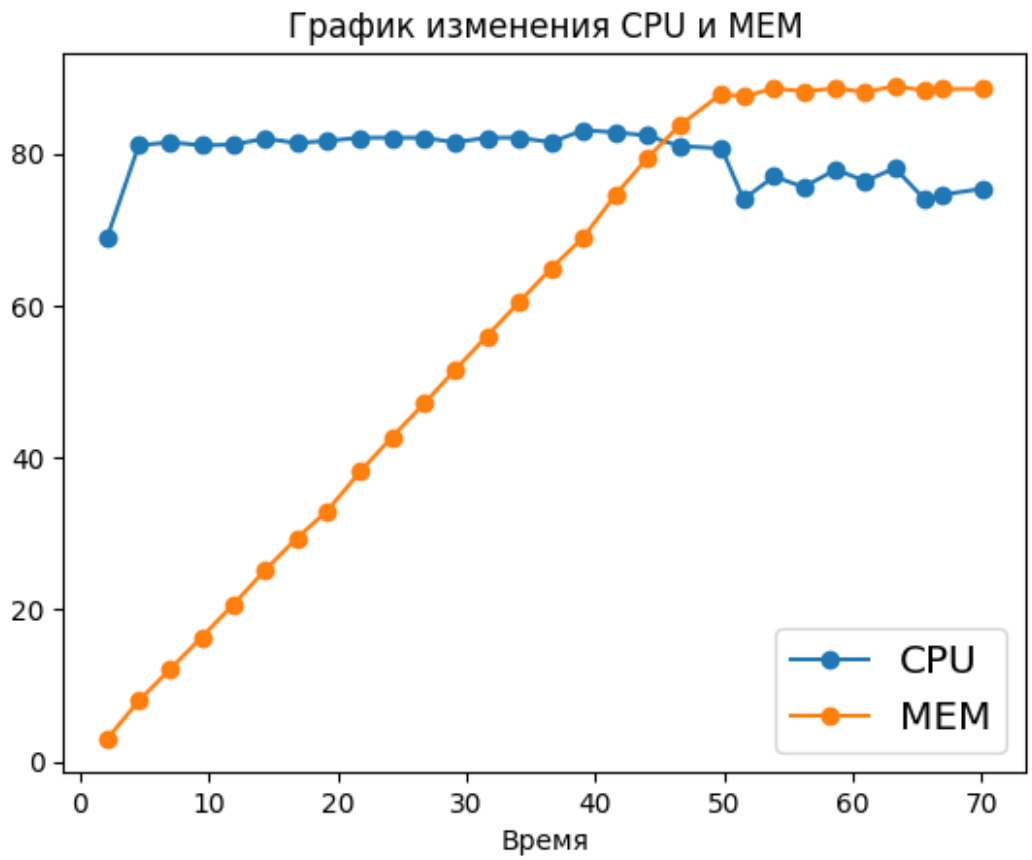
Эксперимент №1

Первый этап

Значения парметров скрипта во время работы:

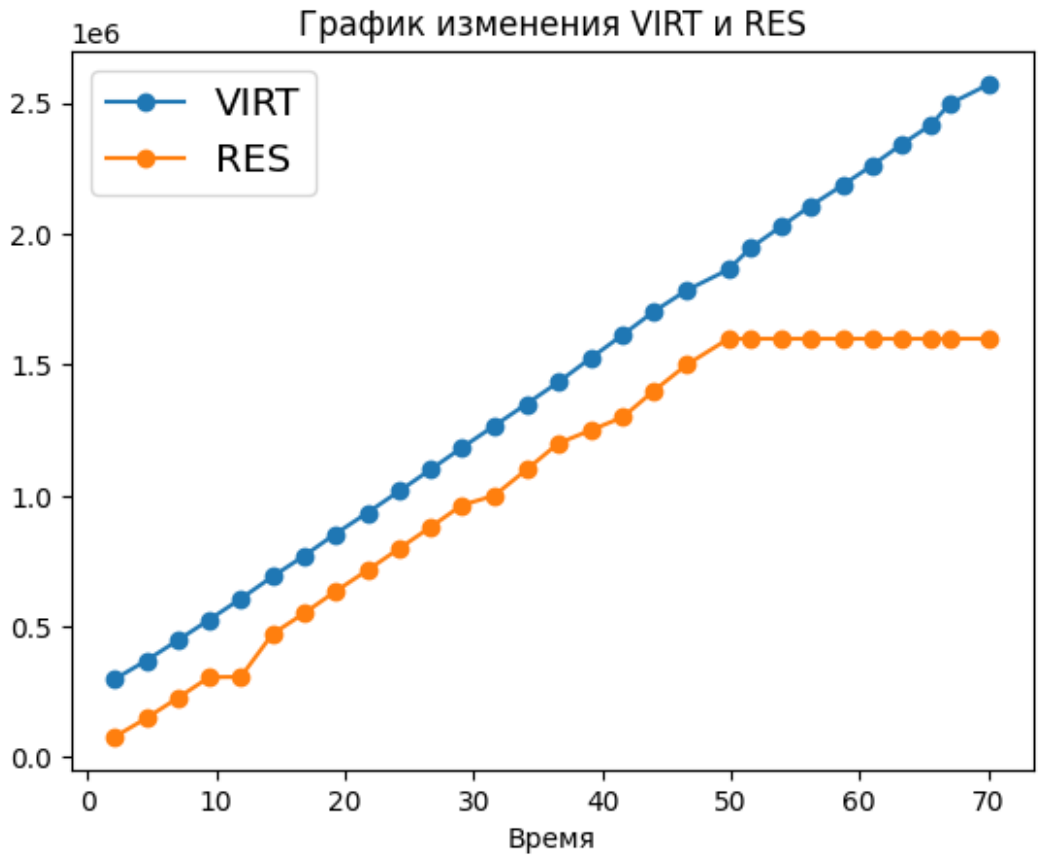
1) PID, USER, PR, NI, SHR, S, COMMAND не изменились 2) TIME+ росло равномерно 3) %MEM, %CPU:

```
import matplotlib.pyplot as plt

cpu = [69, 81.1, 81.5, 81.1, 81.2, 82, 81.4, 81.7, 82.1, 82.1, 82.1,
81.5, 82.1, 82.1,
       81.5, 83.1, 82.8, 82.4, 81.0, 80.7, 74, 77, 75.6, 77.9,  76.4,
78.1, 74, 74.6, 75.4 ]
mem = [3, 8, 12.2, 16.4, 20.7, 25.2, 29.4, 33, 38.2, 42.7, 47.1, 51.5,
56, 60.5, 64.9,
       69, 74.6, 79.3, 83.7, 87.9, 87.5, 88.6, 88.2, 88.6, 88.1, 88.9,
88.3, 88.5, 88.5 ]
time = [2.1, 4.55, 7.01, 9.45, 11.91, 14.37, 16.82, 19.2, 21.76,
24.23, 26.71, 29.1,
        31.6, 34.13, 36.59, 39.1, 41.6, 44, 46.56, 49.83, 51.53, 53.9,
56.2, 58.7, 61, 63.3, 65.6, 67, 70.1 ]
plt.plot(time, cpu, 'o-', label='CPU')
plt.plot(time, mem, 'o-', label='MEM')
plt.title('График изменения CPU и MEM')
plt.xlabel('Время')
plt.legend(fontsize='x-large')
plt.show()
```



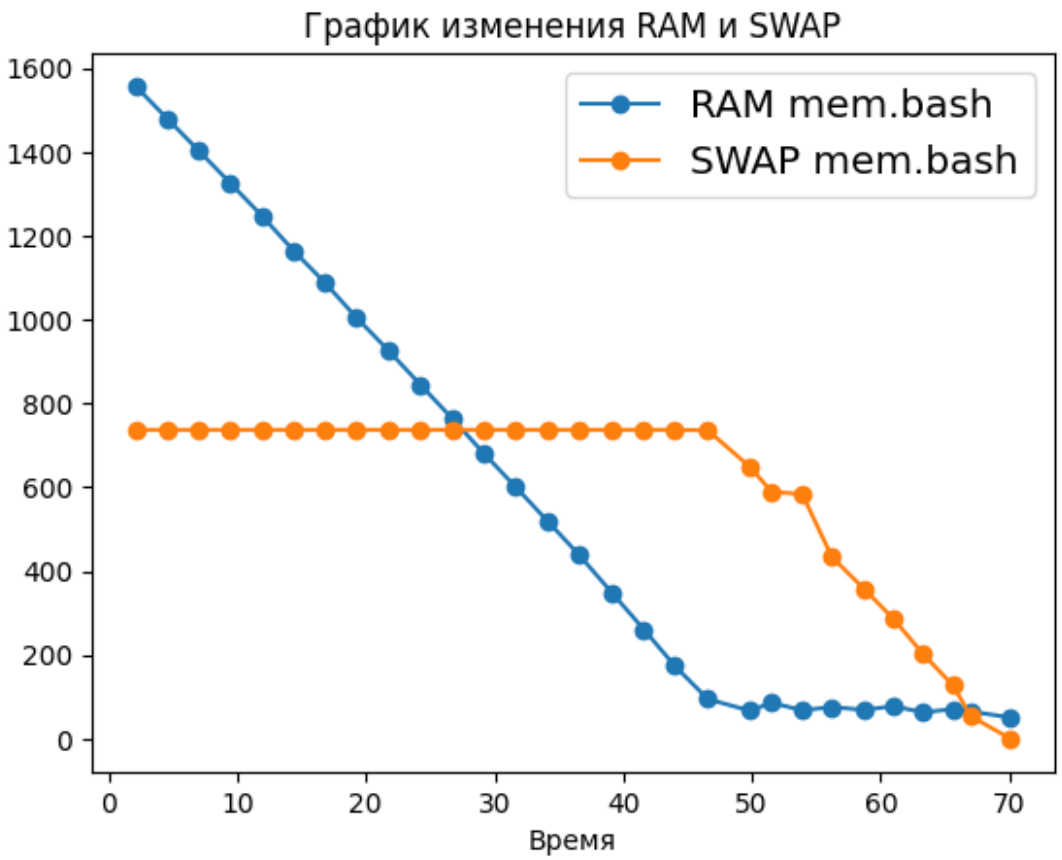

```
4) VIRT, RES:

VIRT = [298872, 369808, 447416, 525956, 606476, 690956, 769892,
852128, 935024, 1017708, 1101808,
        1183712, 1266608, 1350956, 1433720, 1525320,  1614692,
1703264, 1784700, 1864568, 1946672,
        2028308, 2107316, 2186516, 2262608, 2343068, 2418448, 2496900,
2573144]
RES = [78508, 149444, 227052, 306524, 307844, 471524, 550460, 632028,
715460, 798356, 881780,
       964140, 1000000, 1100000, 1200000, 1250000, 1300000, 1400000,
1500000, 1600000, 1600000,
       1600000, 1600000, 1600000, 1600000, 1600000, 1600000, 1600000,
1600000]
time = [2.1, 4.55, 7.01, 9.45, 11.91, 14.37, 16.82, 19.2, 21.76,
24.23, 26.71, 29.1,
        31.6, 34.13, 36.59, 39.1, 41.6, 44, 46.56, 49.83, 51.53, 53.9,
56.2, 58.7, 61, 63.3, 65.6, 67, 70.1 ]
plt.plot(time, VIRT, 'o-', label='VIRT')
plt.plot(time, RES, 'o-', label='RES')
plt.title('График изменения VIRT и RES')
plt.xlabel('Время')
```

```
plt.legend(fontsize='x-large')
plt.show()
```



Free memory

```
ram = [1557, 1480, 1403, 1326, 1248, 1165, 1088, 1007, 926, 845, 764,
683, 602, 519,
       438, 349, 261, 174, 95, 67, 86, 68, 76, 69, 78, 63, 71, 65, 51]
swap = [737, 737, 737, 737, 737, 737, 737, 737, 737, 737, 737, 737,
737, 737,
        737, 737, 737, 737, 736, 649, 589, 584, 434, 358, 286, 202,
128, 54, 0]
time = [2.1, 4.55, 7.01, 9.45, 11.91, 14.37, 16.82, 19.2, 21.76,
24.23, 26.71, 29.1,
        31.6, 34.13, 36.59, 39.1, 41.6, 44, 46.56, 49.83, 51.53, 53.9,
56.2, 58.7, 61, 63.3, 65.6, 67, 70.1]
plt.plot(time, ram, 'o-', label='RAM mem.bash')
plt.plot(time, swap, 'o-', label='SWAP mem.bash')

plt.title('График изменения RAM и SWAP')
plt.xlabel('Время')
plt.legend(fontsize='x-large')
plt.show()
```



Изменения в верхних пяти процессах Изначально было 4 процесса top, kuorker, ksuapd8, systemd, на пятой секунде процесс bash занял первое место и вытеснил top. До остановки скрипта эти процессы иногда появлялись на секунду по mem.sh и надолго исчезали, после остановки процессы вернулись на место

Команда dmesg | grep "mem.bash"

```
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
End step: 300001: size: 3000010
```

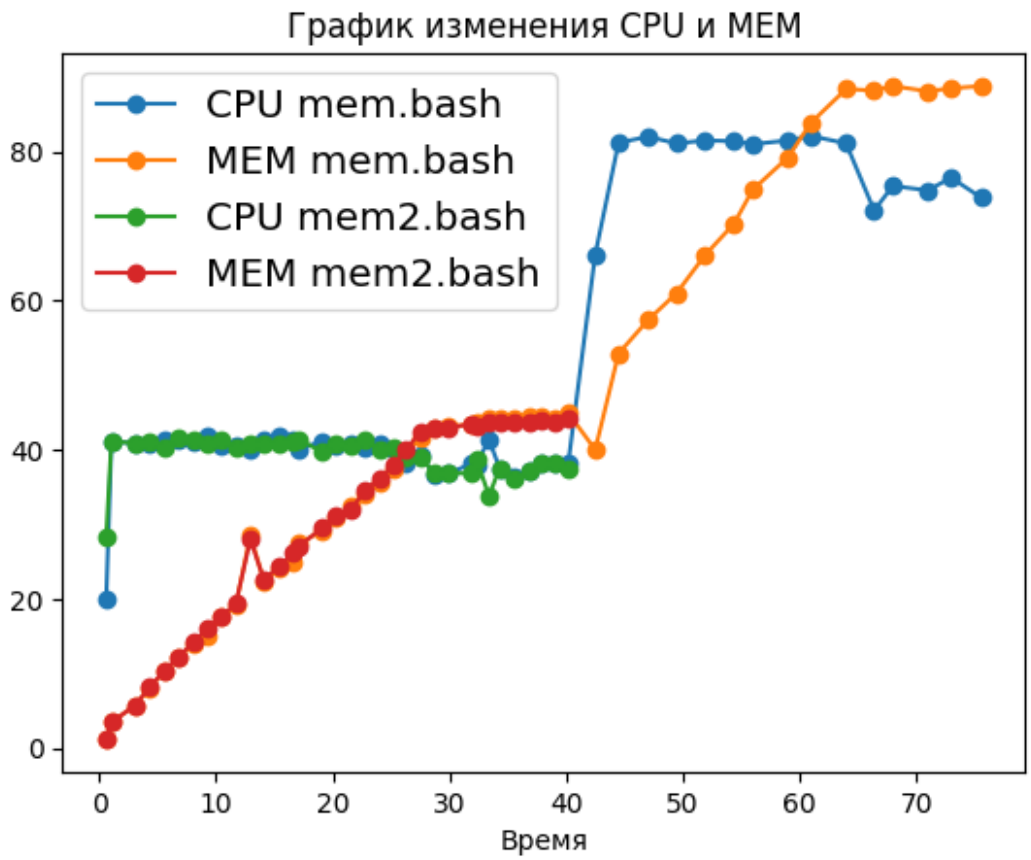
Значение последней строки в файле report.log — 30000000

Второй этап

Значения парметров скрипта во время работы:

1) PID, USER, PR, NI, SHR, S, COMMAND не изменились 2) TIME+ росло равномерно 3) %MEM, %CPU:

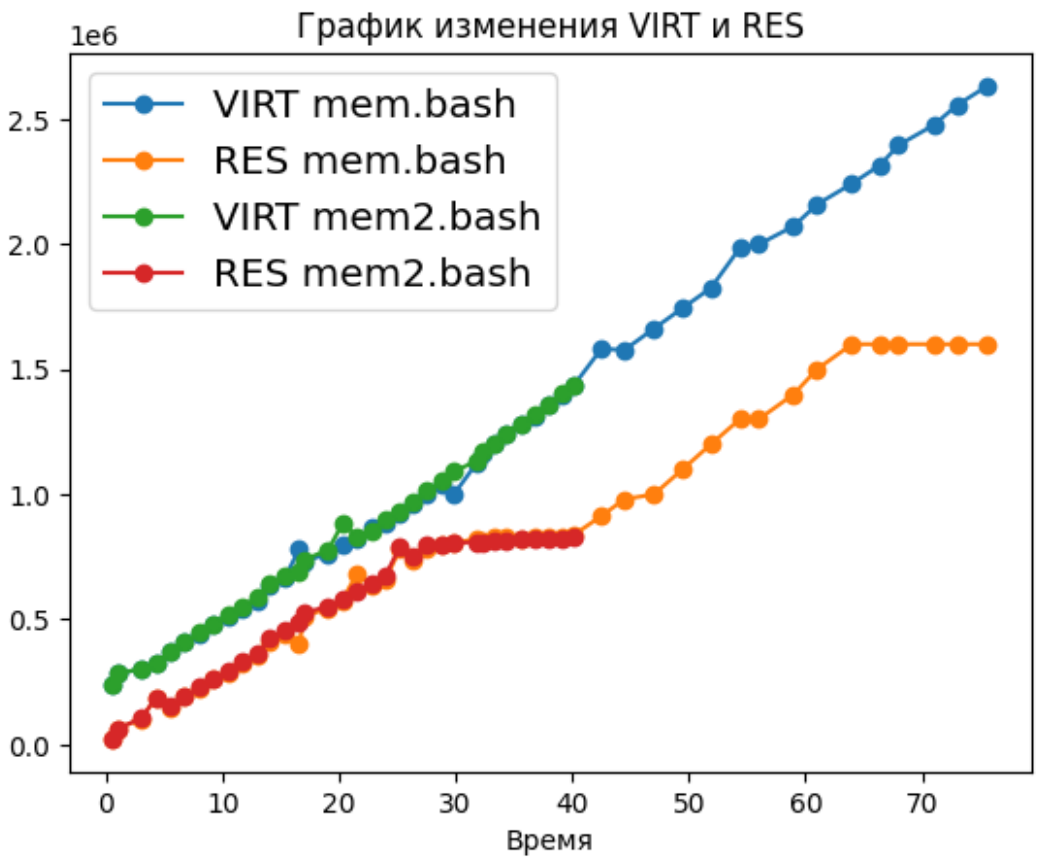
```
cpu1 = [19.9, 41.1, 40.9, 40.7, 41.4, 41.2, 41.1, 41.9, 40.5, 40.5,
39.9, 41.4, 41.8, 41.4, 40.1,
        41.1, 40.5, 40.9, 40.3, 40.7, 39.9, 38.2, 39.3, 36.6, 36.8,
38.1, 38.0, 41.4, 37.5, 36.4,
        37.1, 38.2, 38.2, 38.3, 66.2, 81.2, 82, 81.1, 81.5, 81.4,
81.0, 81.5, 82.1, 81.1,
        72.1, 75.4, 74.8, 76.5, 73.8]
cpu2 = [28.3, 41.1, 40.9, 41.0,  40.4, 41.5, 41.4, 40.9, 41.2, 40.2,
40.9, 40.7, 40.7,
        41.1, 41.4, 39.7, 40.9, 40.5, 41.3, 40.1, 40.3, 38.9, 38.9,
36.9, 36.9, 36.8, 38.8,
        33.8, 37.5, 36.0, 37.1, 38.2, 38.2, 37.3]
mem1 = [1.2, 3.4, 5.7, 8.0, 10.2, 12.1, 13.9, 15, 17.5, 19.1, 28.5,
22.4, 24.0, 25.0, 27.4, 29.1,
        30.8, 32.4, 34.1, 35.7, 37.4, 39.5, 41.7, 42.9, 43.1, 43.4,
43.7, 44.2, 44.3, 44.3, 44.4, 44.4, 44.3, 44.9,
        40.0, 52.9, 57.4, 61.0, 66.1, 70.3, 74.9, 79.2, 83.9, 88.4,
88.2, 88.8, 88.1, 88.5, 88.8]
mem2 = [1.2, 3.4, 5.7, 8.1, 10.3, 12.2, 14.1, 16, 17.7, 19.3, 28.0,
22.6, 24.3, 26.1, 27.0, 29.5,
        31.2, 32.0, 34.5, 36.2, 37.9, 40.1, 42.3, 42.8, 43.0, 43.4,
43.1, 43.7, 43.8, 43.7, 43.8, 43.9, 43.8, 44.3]
time1 = [0.6, 1.04, 3.07, 4.3, 5.54, 6.78, 8.03, 9.27, 10.5, 11.7,
12.95, 14.1, 15.4, 16.6, 17.08, 19.1,
         20.3, 21.56, 22.79, 24.01, 25.22, 26.3, 27.5, 28.75, 29.9,
31.86, 32.31, 33.4, 34.3, 35.6, 36.8,
         37.9, 39.1, 40.2, 42.5, 44.5, 47.02, 49.47, 51.9, 54.38, 56,
59, 61, 64, 66.4, 68, 71, 73, 75.6]
time2 = [0.6, 1.04, 3.07, 4.3, 5.54, 6.78, 8.03, 9.27, 10.5, 11.7,
12.95, 14.1, 15.4, 16.6, 17.08, 19.1,
         20.3, 21.56, 22.79, 24.01, 25.22, 26.3, 27.5, 28.75, 29.9,
31.86, 32.31, 33.4, 34.3, 35.6, 36.8,
         37.9, 39.1, 40.2]
plt.plot(time1, cpu1, 'o-', label='CPU mem.bash')
plt.plot(time1, mem1, 'o-', label='MEM mem.bash')
plt.plot(time2, cpu2, 'o-', label='CPU mem2.bash')
plt.plot(time2, mem2, 'o-', label='MEM mem2.bash')
plt.title('График изменения CPU и MEM')
plt.xlabel('Время')
plt.legend(fontsize='x-large')
plt.show()
```



4) VIRT, RES:

```
virt1 = [242156, 282416, 302518, 325316, 368348, 410720, 444908,
479096, 512660, 546152, 576240,
         637092, 667724, 782844, 732668, 763020, 795104, 825200,
865136, 887372, 919844, 958520,
         1001024, 1041020, 1002336, 1128080, 1161176, 1202588,
1240340, 1278620, 1314656, 1354652,
         1394304, 1438824, 1581032, 1577336, 1662476, 1744316,
1824440, 1984564, 1998628, 2071816,
         2158664, 2241692, 2315340, 2395736, 2475228, 2555060,
2631628]
virt2 = [242552, 283076, 303518, 326636, 370468, 412436, 447012,
482660, 522660, 550640, 588472, 643040, 673400,
         688240, 739532, 778684, 882364, 832724, 856748, 895952,
929612, 969344, 1012244, 1052900,
         1093608, 1133156, 1167862, 1206204, 1244960, 1283504,
1320596, 1361516, 1400456, 1435832]
res1 = [22848, 63240, 103233, 186880, 149040, 191544, 225600, 259656,
284336, 326712, 357072, 410594,
        440416, 402736, 513136, 544512, 575928, 685760, 637440,
660864, 780180, 738872, 779416, 801984,
        806764, 822284, 816752, 827300, 829040, 820480, 830816,
```

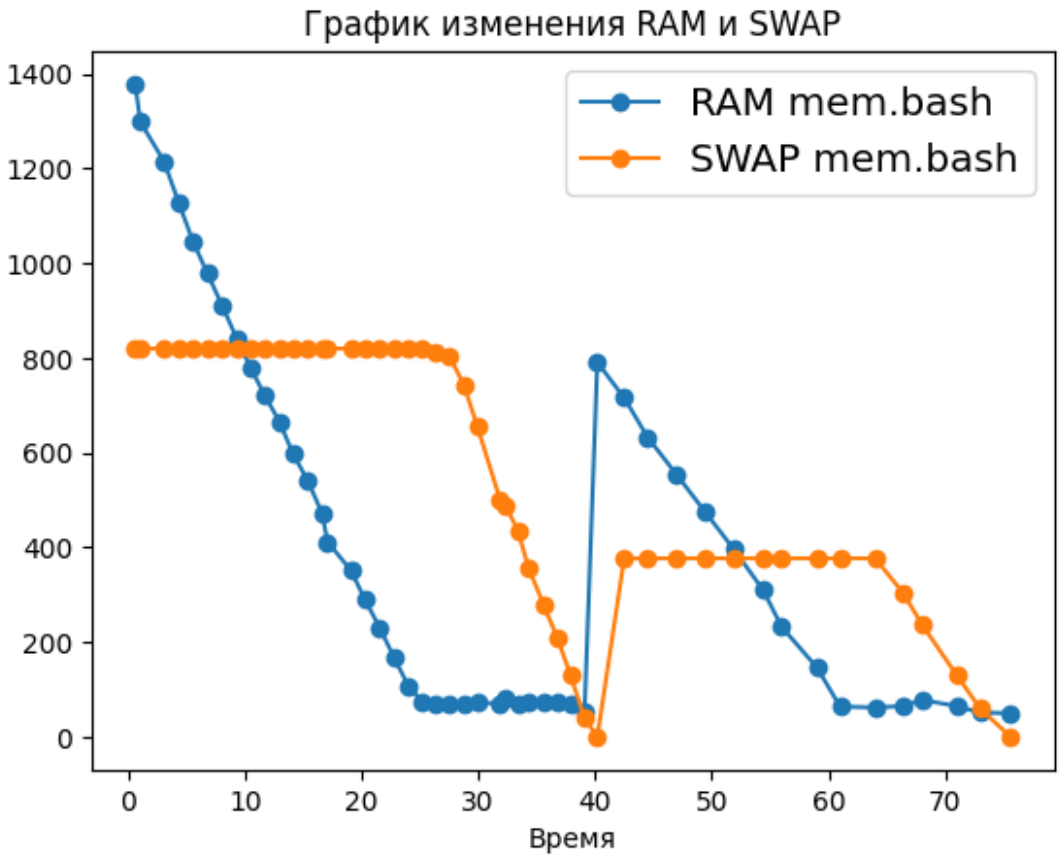
```
830964, 829216, 839624, 913276, 980780,
        1000000, 1100000, 1200000, 1300000, 1300000, 1400000, 1500000,
1600000, 1600000, 1600000, 1600000,
        1600000, 1600000]
res2 = [23096, 63753, 105233, 187312, 151136, 193112, 228200, 263336,
294336, 331184, 361816, 423504, 453944, 488792,
        528200, 551360, 583840, 613480, 646136, 676760, 789892,
749740, 798824, 801464, 805268, 809872,
        806124, 816928, 810672, 818232, 819524, 821124, 818560,
828124]
time1 = [0.6, 1.04, 3.07, 4.3, 5.54, 6.78, 8.03, 9.27, 10.5, 11.7,
12.95, 14.1, 15.4, 16.6, 17.08, 19.1,
         20.3, 21.56, 22.79, 24.01, 25.22, 26.3, 27.5, 28.75, 29.9,
31.86, 32.31, 33.4, 34.3, 35.6, 36.8,
         37.9, 39.1, 40.2, 42.5, 44.5, 47.02, 49.47, 51.9, 54.38, 56,
59, 61, 64, 66.4, 68, 71, 73, 75.6]
time2 = [0.6, 1.04, 3.07, 4.3, 5.54, 6.78, 8.03, 9.27, 10.5, 11.7,
12.95, 14.1, 15.4, 16.6, 17.08, 19.1,
         20.3, 21.56, 22.79, 24.01, 25.22, 26.3, 27.5, 28.75, 29.9,
31.86, 32.31, 33.4, 34.3, 35.6, 36.8,
         37.9, 39.1, 40.2]
plt.plot(time1, virt1, 'o-', label='VIRT mem.bash')
plt.plot(time1, res1, 'o-', label='RES mem.bash')
plt.plot(time2, virt2, 'o-', label='VIRT mem2.bash')
plt.plot(time2, res2, 'o-', label='RES mem2.bash')

plt.title('График изменения VIRT и RES')
plt.xlabel('Время')
plt.legend(fontsize='x-large')
plt.show()
```



Free memory

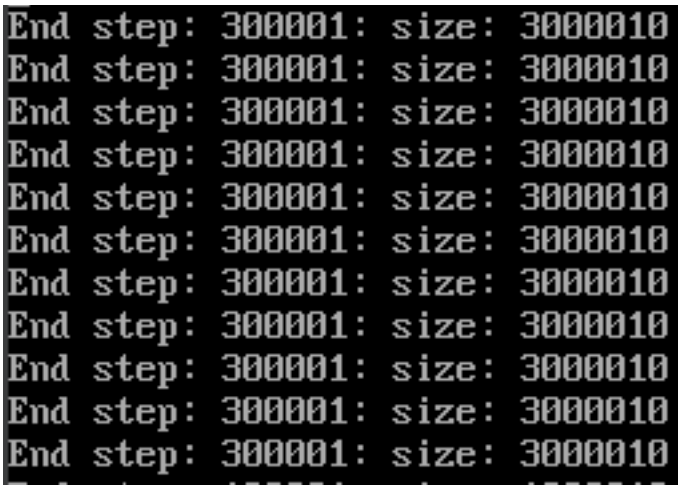
```
ram = [1378.7, 1299, 1215, 1130, 1047, 979, 911.9, 842.8, 779, 721,
666, 599, 540, 473, 412, 351, 289,
       230, 167, 107, 74, 70, 71, 69, 75, 68, 81, 71, 75, 73, 72, 68,
51, 791, 716, 633, 553, 474, 396,
       311, 232, 147, 64, 62, 65, 78, 65, 51, 50]
swap = [820, 820, 820, 820, 820, 820, 820, 820, 820, 820, 820, 820,
820, 820, 820, 820, 820, 820, 820,
        820, 819, 811, 805, 744, 657, 501, 490, 435, 357, 279, 207,
130, 40, 0, 377, 377, 377, 377,
        377, 377, 377, 377, 377, 377, 304, 238, 130, 60, 0]
time = [0.6, 1.04, 3.07, 4.3, 5.54, 6.78, 8.03, 9.27, 10.5, 11.7,
12.95, 14.1, 15.4, 16.6, 17.08, 19.1,
         20.3, 21.56, 22.79, 24.01, 25.22, 26.3, 27.5, 28.75, 29.9,
31.86, 32.31, 33.4, 34.3, 35.6, 36.8,
         37.9, 39.1, 40.2, 42.5, 44.5, 47.02, 49.47, 51.9, 54.38, 56,
59, 61, 64, 66.4, 68, 71, 73, 75.6]
plt.plot(time, ram, 'o-', label='RAM mem.bash')
plt.plot(time, swap, 'o-', label='SWAP mem.bash')

plt.title('График изменения RAM и SWAP')
plt.xlabel('Время')
```

```
plt.legend(fontsize='x-large')
plt.show()
```



Изменения в верхних пяти процессах Изначально было 4 процесса top, kuorker, ksuapd8, systemd, на пятой секунде процесс bash занял первое место и вытеснил top. До остановки скрипта эти процессы иногда появлялись на секунду по mem.sh и надолго исчезали, после остановки процессы вернулись на место

Команда dmesg | grep "mem.bash"



Значение последней строки в файле report.log — 30000000

Значение последней строки в файле report.log — 15000000

Что получили?

```
1) %CPU держится высоким, так как процесс просит только дополнительную память. Поэтому неполная загрузка процессора процессами и падение загрузки перед аварийной остановкой.

2) %MEM используемый объем памяти растёт. При использовании файла подкачки %MEM останавливается и держится в одном диапазоне, так как часть страниц памяти, используемых процессом, переносится в swap. Страницы переносятся не по одной, поэтому происходят скачки. Аналогичное верно про RES.

3) В случае запуска одного скрипта количество элементов массива при аварийной остановке = 30.000.000. Total Memory при аварии = 2.644.424Kb. То есть 90 байт на один элемент.

4) При запуске двух скриптов второй скрипт останавливается при использовании примерно половины от Total Memory. Так как первый скрипт использует столько же и в сумме используется вся доступная память. После остановки второго скрипта первый останавливается при всем Total Memory, так как теперь память использует только он и фоновые процессы.
```

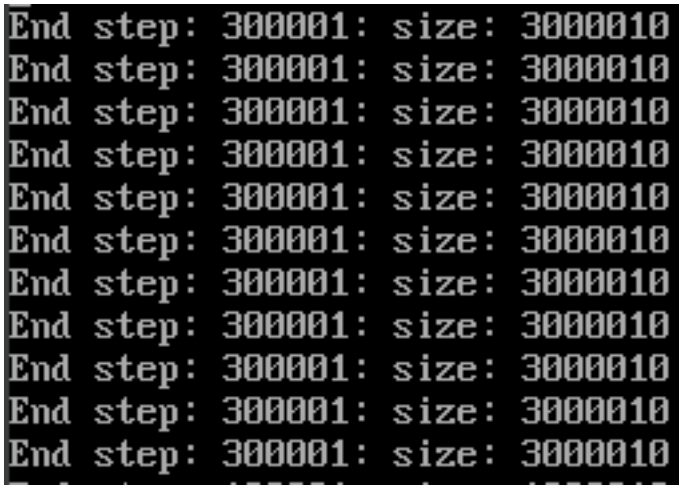
Вывод: было проанализировано поведение механизмов управления памятью OS Linux при растущем использовании памяти. Теоретические сведения были подтверждены двумя эксперементами.

Эксперимент №2

K=10

N=30.000.000 / 10

Прошли успешно. Так как разбили работу, которую выполнял один "поток" на 10 "потоков".



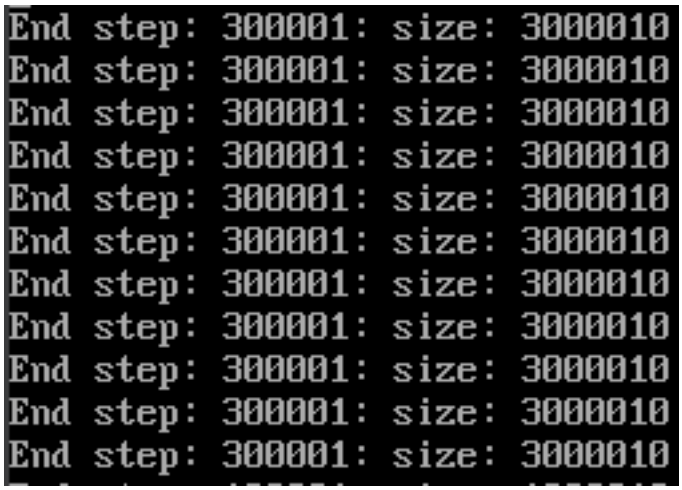
K=30

N=30.000.000 / 10

11 прошли успешно 19 провалились

Так как раньше выполнял один поток 30.000.000; А теперь 30 потоков пытаются выполнить 3.000.000 * 30 = 90.000.000, в итоге приблизительно в 3 раза больше, очевидно, что не справится.

Но как можно заметить. Сделали на 1.000.000 больше чем при K=1. Почему? Так как мы разбили нашу работу и работаем на самом деле не параллельно а конкурентно последовательно, поэтому раюота разбивается на несколько и может выполнится больше.



K=30

N=30.000.000 / 10 / 3

30 прошли успешно

Очевидно, просто разбили работу на 30 "потоков"

K=30

N=30.000.000 / 10 / 3 + 40.000

30 прошли успешно

Максимальное значение N, такое что при K=30 не происходило аварийных остановок процессов - 1.000.000 + 40.000. Оно отличается от ожидаемого значения - 1.000.000, так как процессы не работают синхронно и некоторые достигают штатного завершения раньше других, пока оперативной памяти хватает