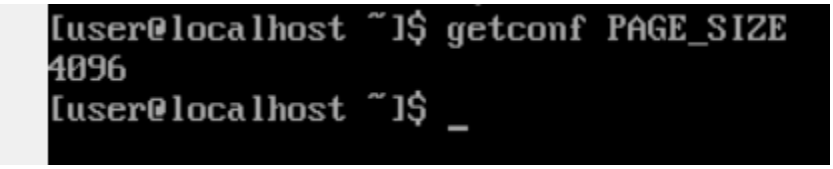
Лабораторная работа №5. Управление памятью в ОС Linux

Начальная конфигурация системы:

- PageSize: 4096 B



- MemTotal: 1870900 kB

- SwapTotal: 839676 kB

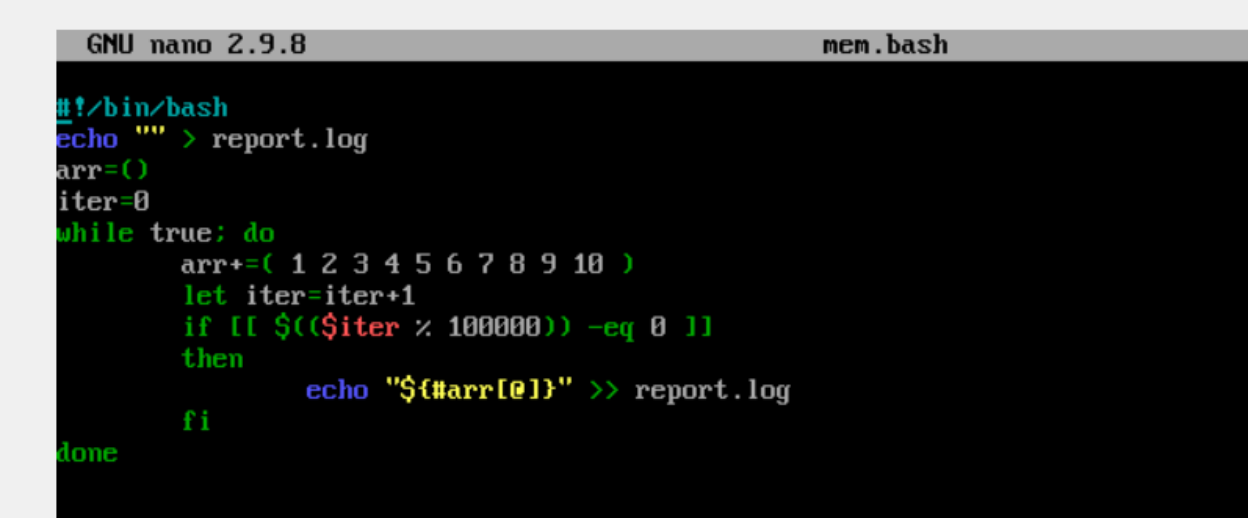
- MemFree: 1234708 kB

- SwapFree: 839676 kB

Эксперимент №1

Первый этап

-Скрипт : mem.bash

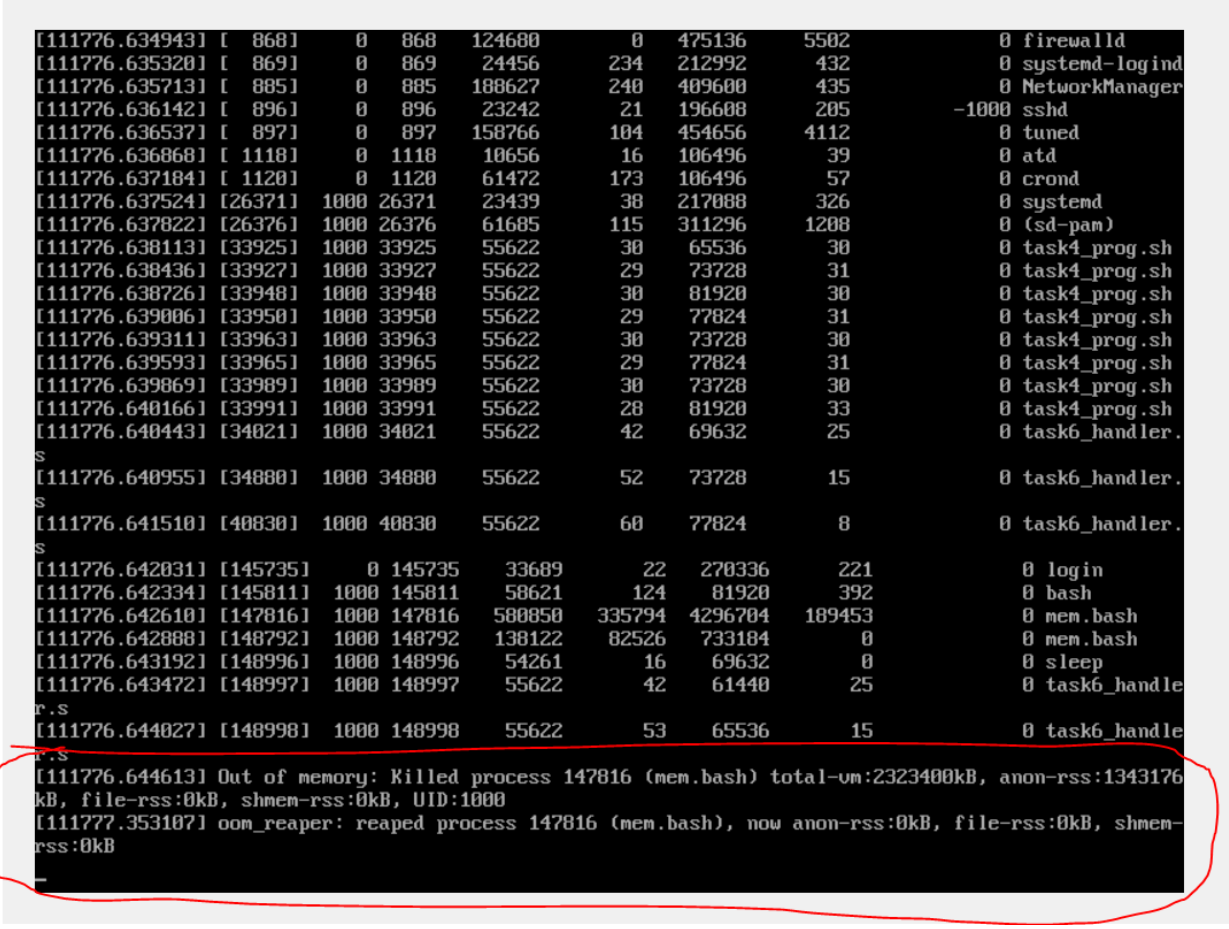


-Скрипт слежения: watch1.bash

```
#!/bin/bash
echo "" > first_five_proces1.txt
echo "" > proc_membash1.txt
echo "Time" > mem1.txt
echo "Time" > swap1.txt

while true; do
        proc_exist=$(top -b -n 1 | grep "mem.bash")
        st=$(echo "$proc_exist" | awk '{print $8}')
        if [[ -z proc_exist || "$st" != "R" ]]; then
                exit 0
        fi
        mem_and_swap=$(top -b -n 1 | head -n5 | tail -n2)
        mem=$(echo "mem_and_swap" | head -n1)
        swap=$(echo "mem_and_swap" | tail -n1)
        first_five=$(top -b -n 1 | head -n12 | tail -n5)
        echo -e "\n" >> first_five_proces1.txt
        proc_exist=$(top -b -n 1 | grep "mem.bash")
        cur_time=$(echo "proc_exist" | awk '{print $11}')
        echo "$cur_time $proc_exist" >> proc_membash1.txt
        echo "$cur_time $mem" >> mem1.txt
        echo "$cur_time $swap" >> swap1.txt
        sleep 1
done
```

-Скрипт запуска: run1.bash

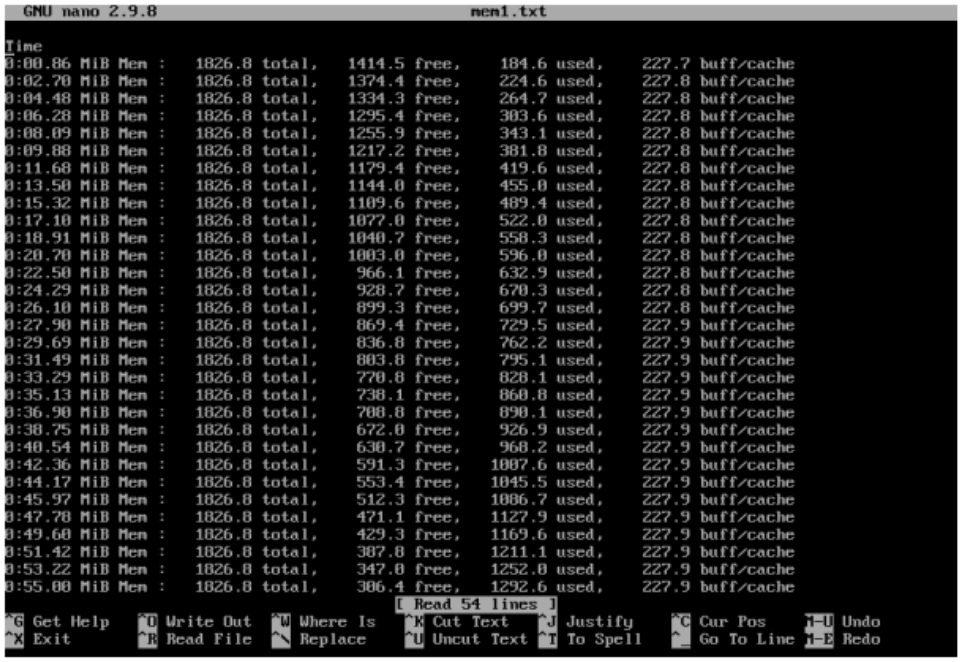
-Журнал при втором запуске mem.bash:



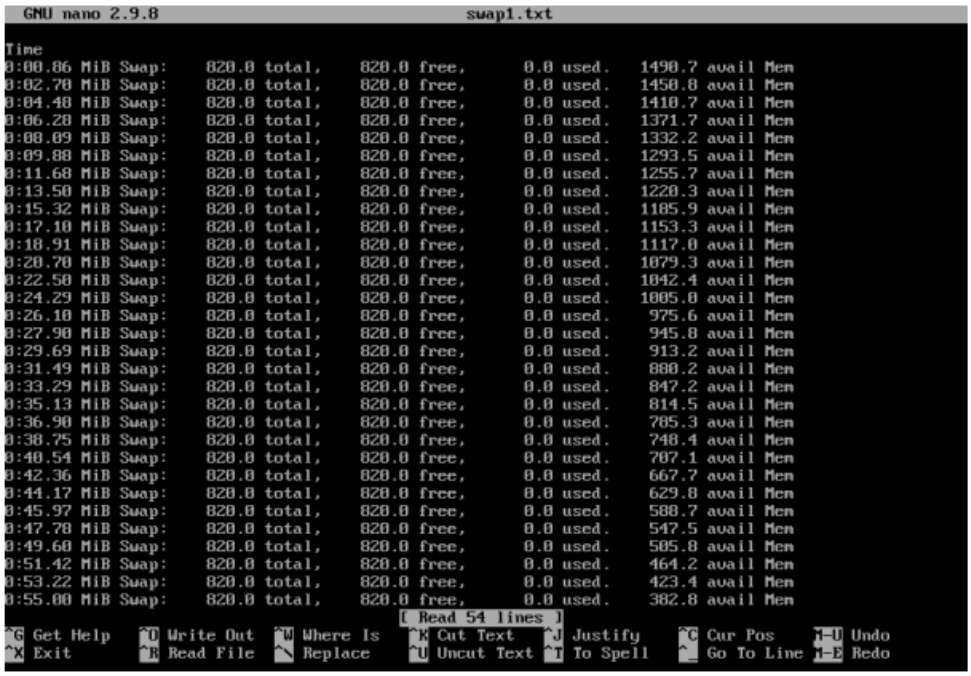
[111776.644613] Out of memory: Killed process 147816 (mem.bash) total-vm:2323400 kB, anon-rss: 1343176 kB, file-rss: 0 kB, shmem-rss:0kb, UID: 1000

[111777.353107] oom_reaper: reaped process 147816 (mem.bash), now anon-rss: 0 kB, file-rss 0 kB, shmem-rss: 0 kB

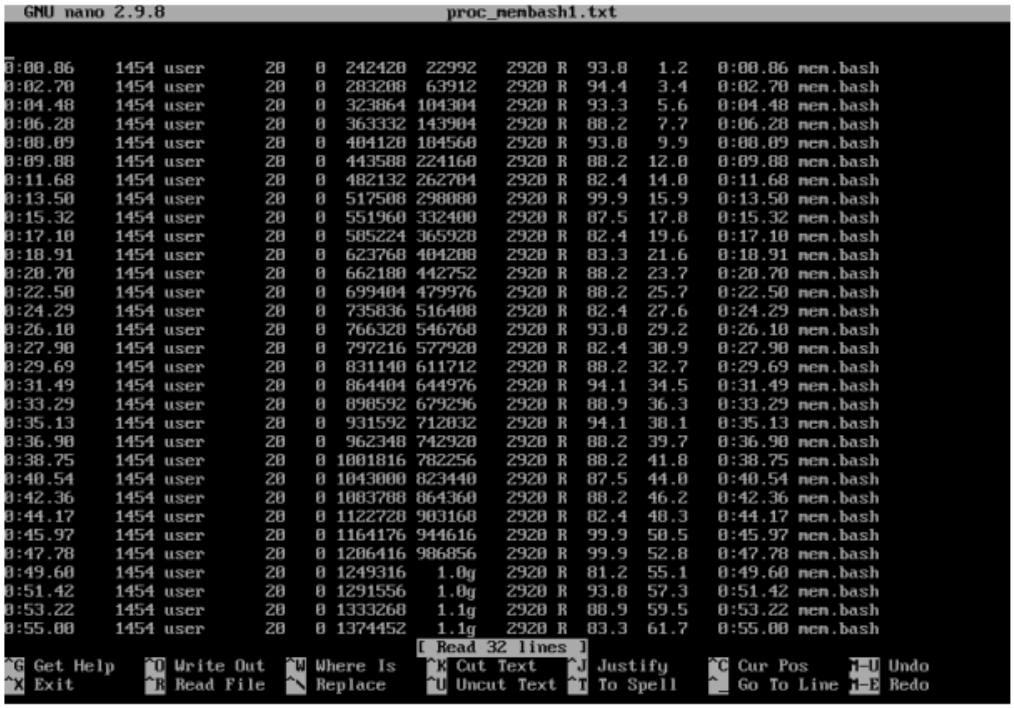
-Файл отслеживания памяти: mem1.txt



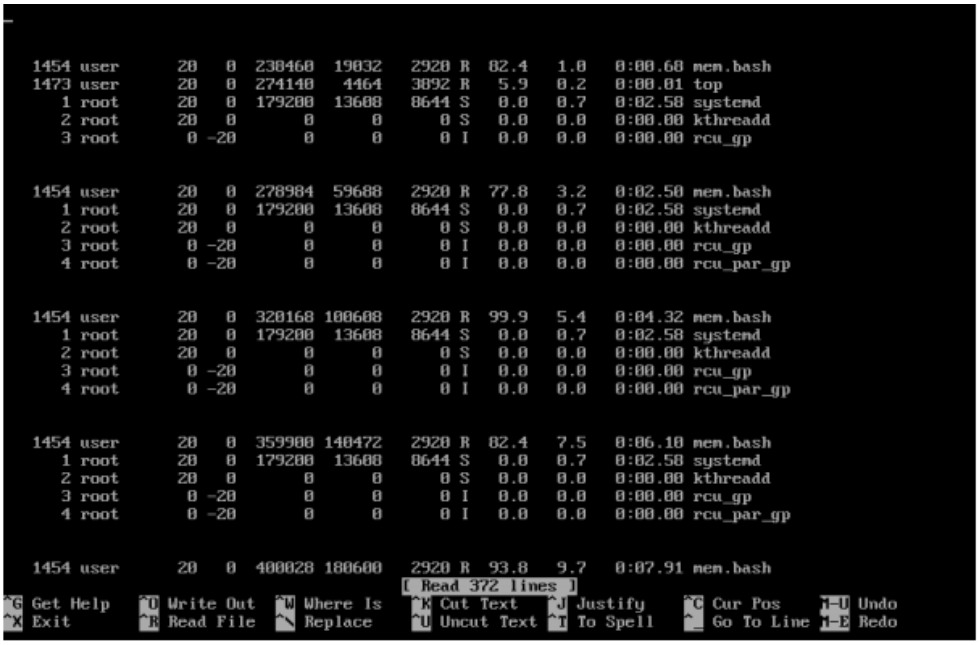
-Файл отслеживания памяти раздела подкачки (Swap): swap1.txt



-Файл отслеживания параметров процесса mem.bash: proc_membash1.txt



Файл отслеживания первых пяти процессов top: first_five_process1.txt



Последняя строка report.log: 30000000

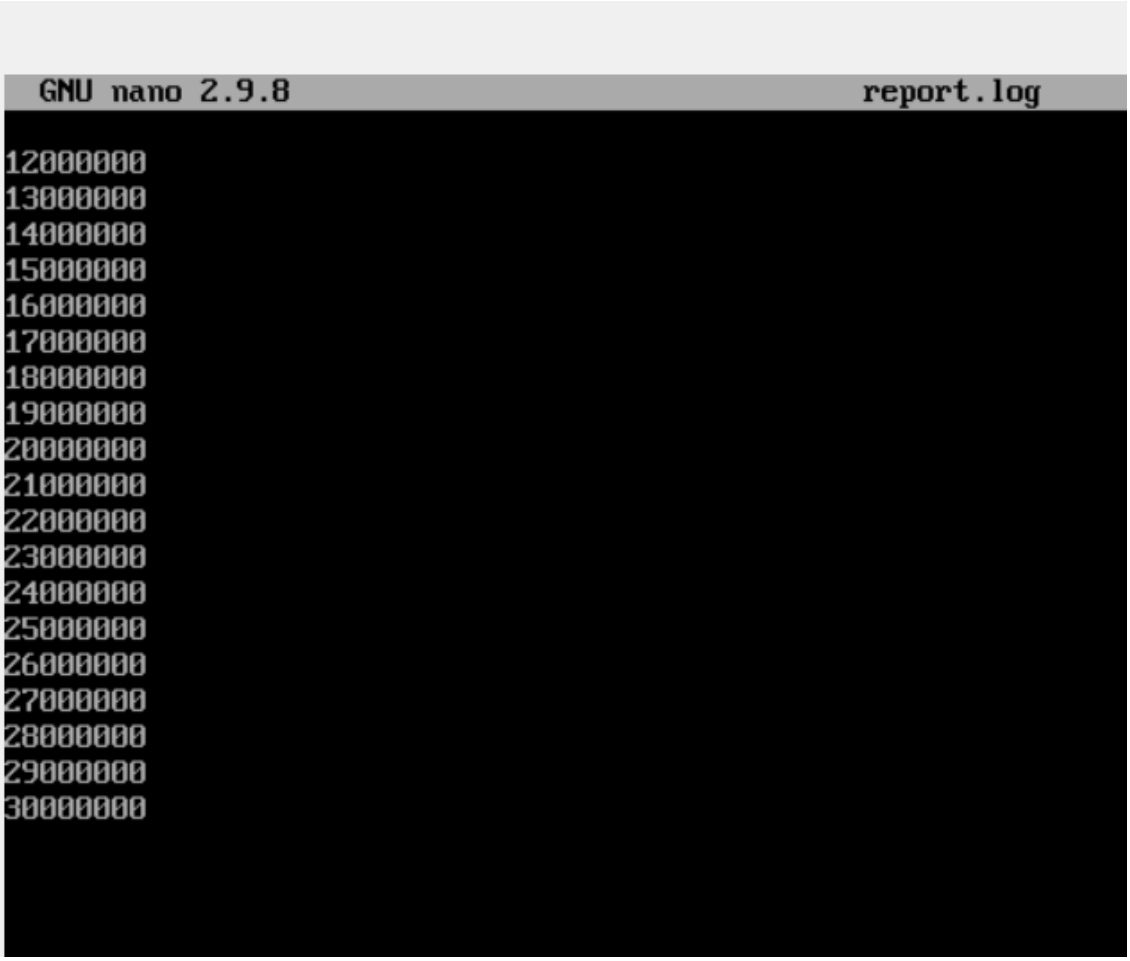
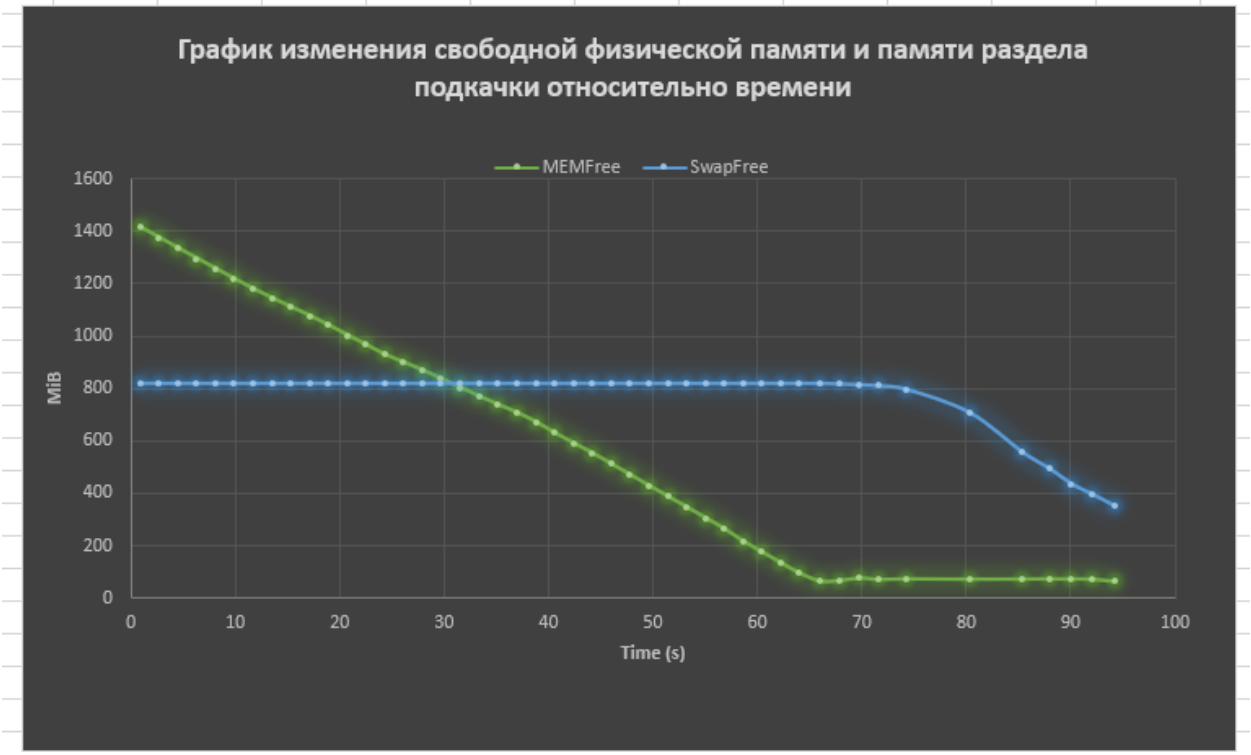


График изменения свободной физической памяти и памяти раздела подкачки относительно времени (Построение из значении файлов mem1.txt и swap1.txt)



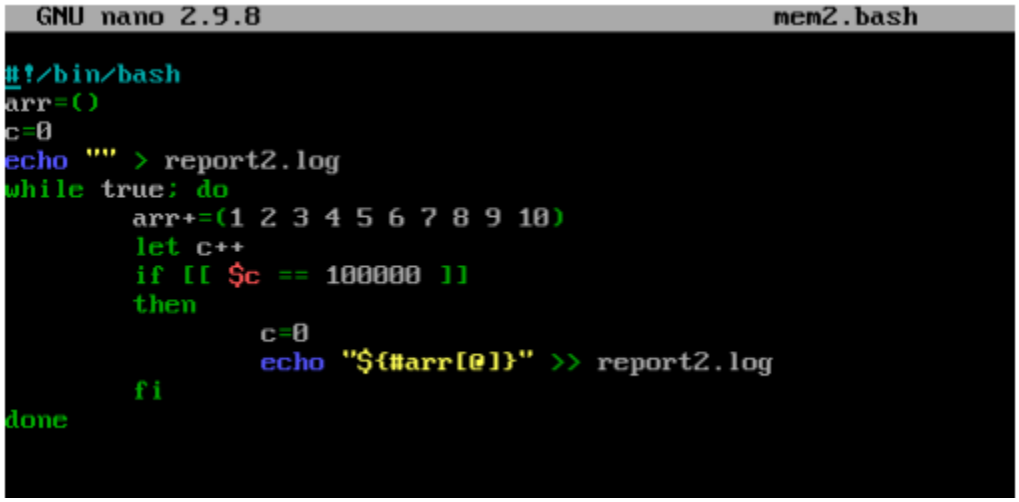
Выводы

В начале программа занимает только физическую память. Как можно увидеть по графику, свободная физическая память линейно убывает, а размер свободной памяти раздела подкачки остается постоянным. Однако в определенный момент времени (42-43с) свободная физическая память заканчивается, и начинается использование раздела подкачки. Когда же заканчивается и он, программа аварийно завершается.

Размер массива в момент аварийного завершения программы: 30 000 000

Второй этап:

Второй скрипт: mem2.bash



Скрипт слежения: w2.bash

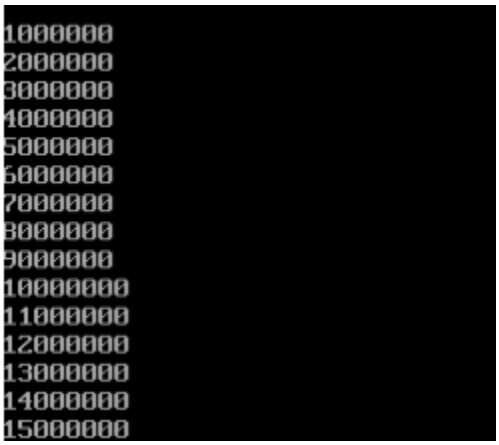
```
#!/bin/bash
echo "" > five_process2.txt
echo "" > proc_membash2.txt
echo "Time" > mem2.txt
echo "Time" > swap2.txt

while true; do
        procExist=$(top -b -n 1 | grep "mem2.bash")
        st=$(echo "$procExist" | awk '{print $8}')
        if [[ -z procExist || "$st" != "R" ]]; then
                exit 0
        fi
        mem_and_swap=$(top -b -n1 | head -n5 | tail -n2)
        mem=$(echo "$mem_and_swap" | head -n1)
        swap=$(echo "$mem_and_swap" | tail -n1)
        five_process=$(top -b -n 1 | head -n12 | tail -n5)
        echo -e "\n" >> five_process2.txt
        procExist=$(top -b -n 1 | grep "mem2.bash")
        cur_time=$(echo "$procExist" | awk '{print $11}')
        echo "$cur_time $procExist" >> proc_membash2.txt
        echo "$cur_time $mem" >> mem2.txt
        echo "$cur_time $swap" >> swap2.txt
        sleep 1
done
```

Скрипт запуска: run2.bash

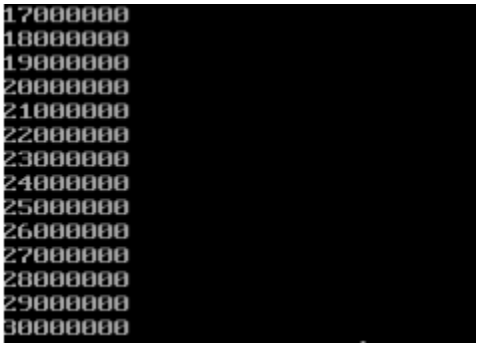
-Журнал при втором запуске

```
[  537.569066] Out of memory: Killed process 1463 (mem.bash) total-vm:2662904kB, anon-rss:1679568kB,
 file-rss:0kB, shmem-rss:0kB, UID:1000
[  537.728687] oom_reaper: reaped process 1463 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss
:0kB
Killed
```

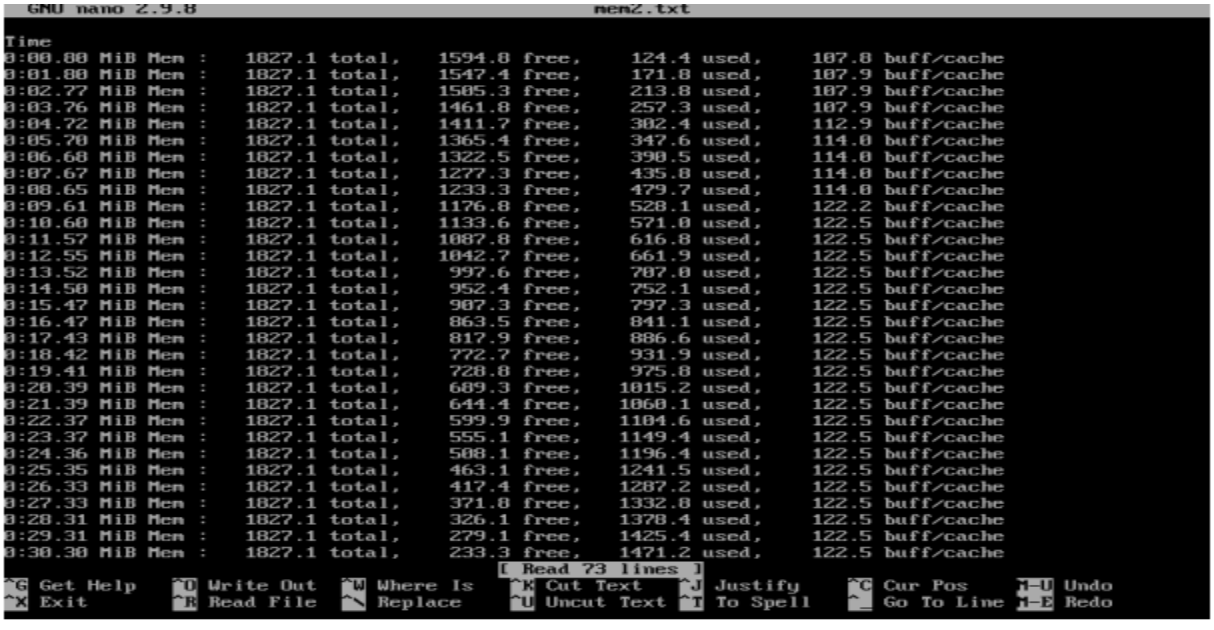
Последняя строка report.log: 15000000



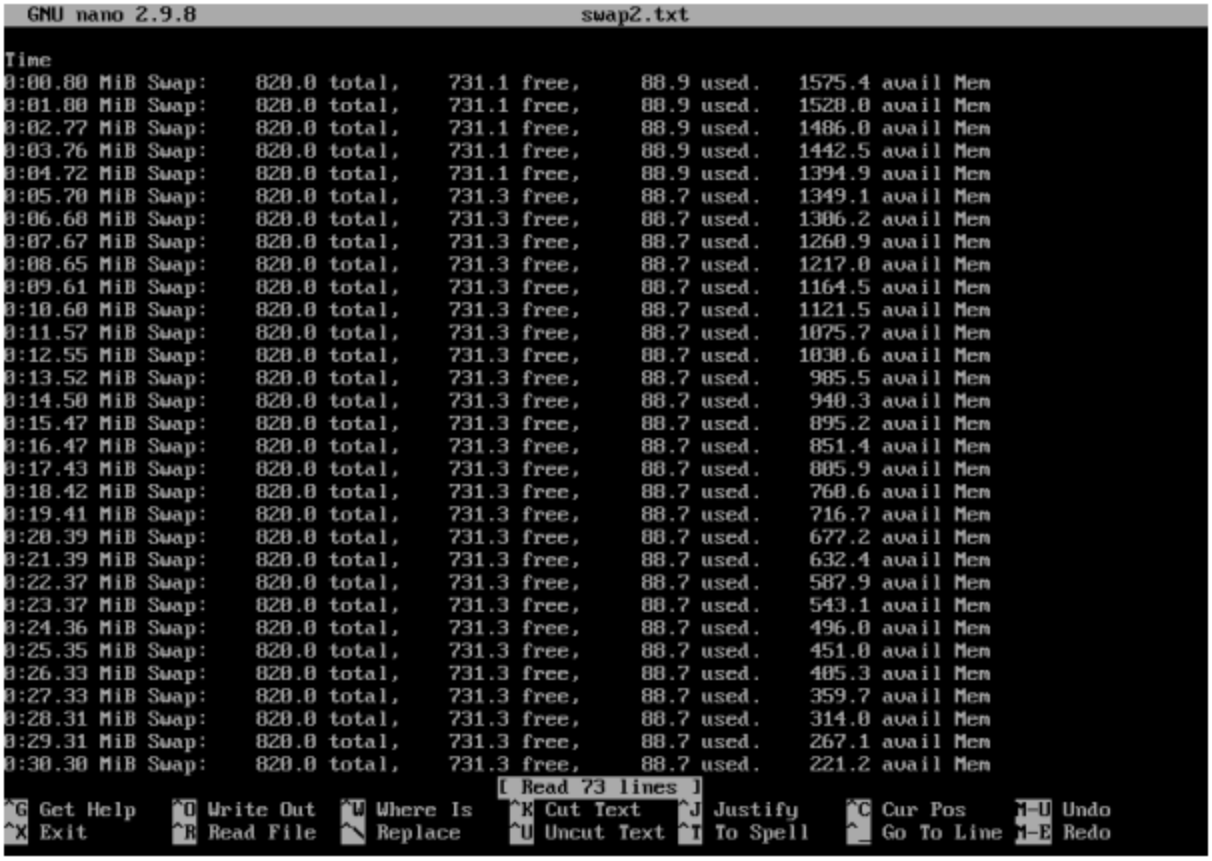
Последняя строка report2.log: 30000000



Файл отслеживания памяти: mem2.txt



Файл отслеживания памяти раздела подкачки (Swap): swap2.txt



Файл отслеживания параметров процесса mem.bash: proc_membash2.txt

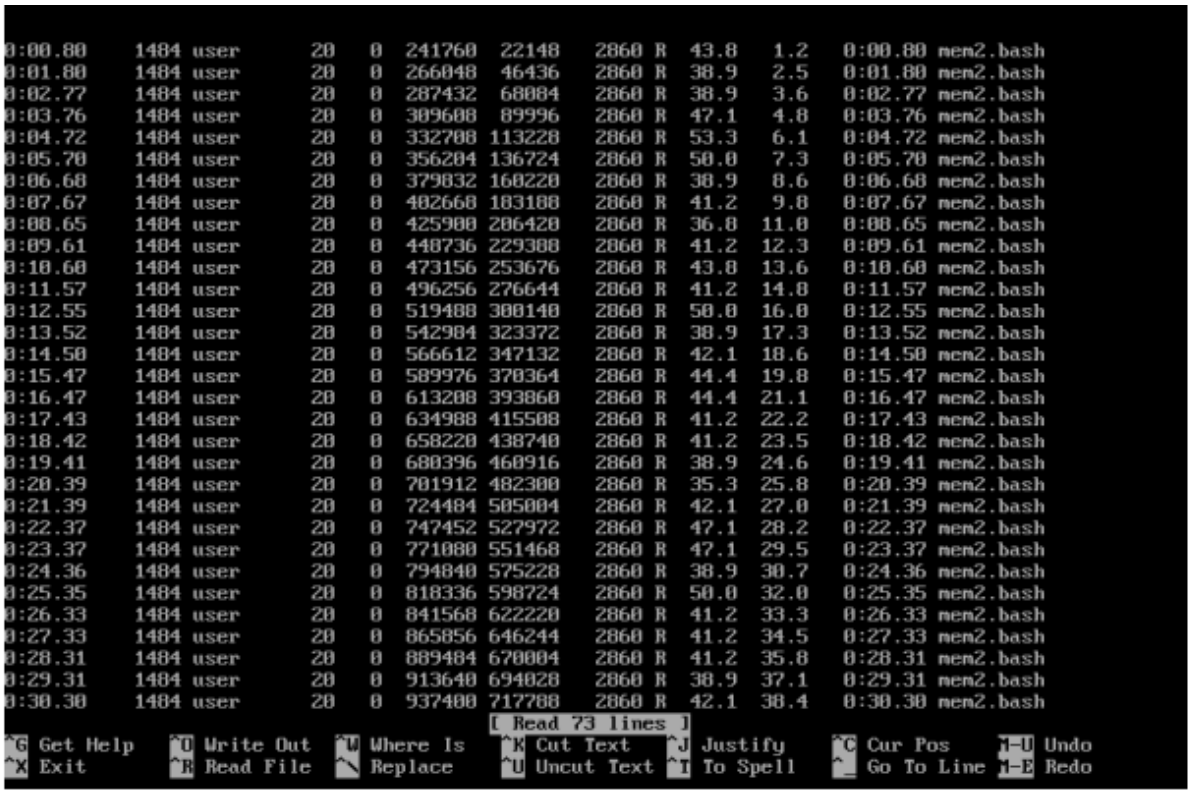
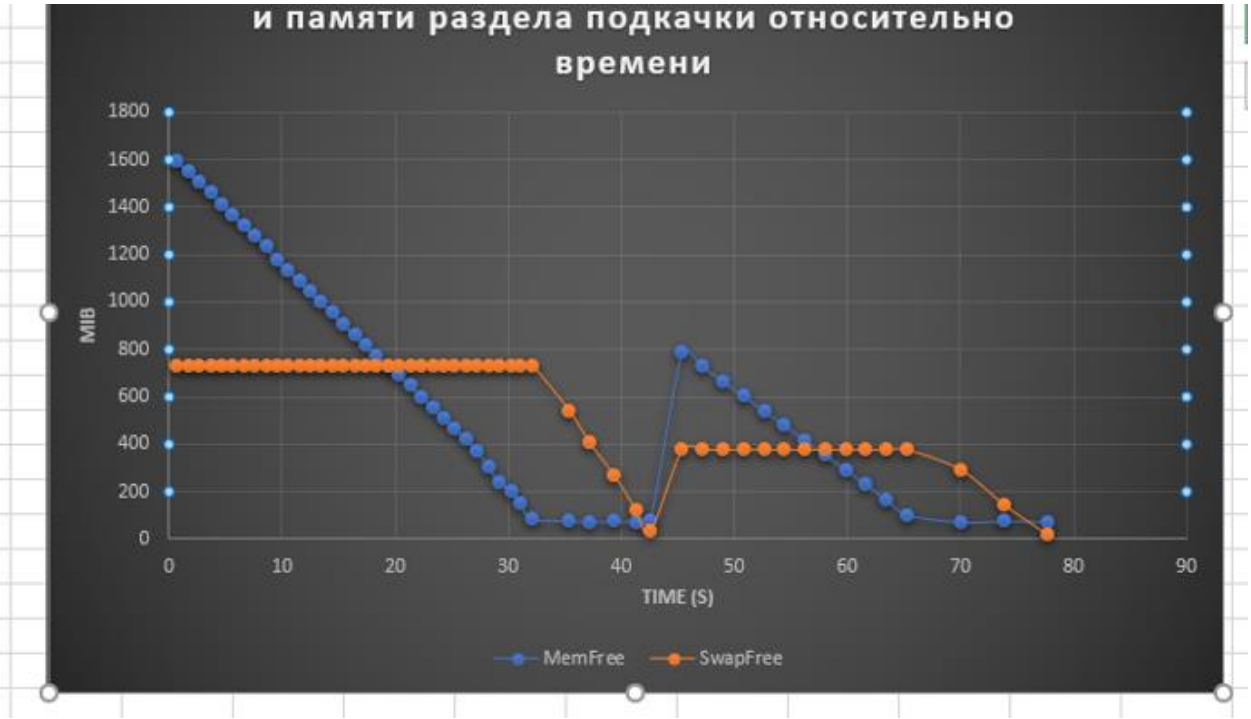

0:00.80 1484 user 20 0 241760 22148 2860 R 43.8 1.2 0:00.80 mem2.bash
0:01.80 1484 user 20 0 266048 46436 2860 R 38.9 2.5 0:01.80 mem2.bash
0:02.77 1484 user 20 0 287432 68084 2860 R 38.9 3.6 0:02.77 mem2.bash
0:03.76 1484 user 20 0 309608 89996 2860 R 47.1 4.8 0:03.76 mem2.bash
0:04.72 1484 user 20 0 332708 113228 2860 R 53.3 6.1 0:04.72 mem2.bash
0:05.70 1484 user 20 0 356204 136724 2860 R 50.0 7.3 0:05.70 mem2.bash
0:06.68 1484 user 20 0 379832 160220 2860 R 38.9 8.6 0:06.68 mem2.bash
0:07.67 1484 user 20 0 402668 183188 2860 R 41.2 9.8 0:07.67 mem2.bash
0:08.65 1484 user 20 0 425900 206420 2860 R 36.8 11.0 0:08.65 mem2.bash
0:09.61 1484 user 20 0 448736 229388 2860 R 41.2 12.3 0:09.61 mem2.bash
0:10.60 1484 user 20 0 473156 253676 2860 R 43.8 13.6 0:10.60 mem2.bash
0:11.57 1484 user 20 0 496256 276644 2860 R 41.2 14.8 0:11.57 mem2.bash
0:12.55 1484 user 20 0 519488 300140 2860 R 50.0 16.0 0:12.55 mem2.bash
0:13.52 1484 user 20 0 542984 323372 2860 R 38.9 17.3 0:13.52 mem2.bash
0:14.50 1484 user 20 0 566612 347132 2860 R 42.1 18.6 0:14.50 mem2.bash
0:15.47 1484 user 20 0 589976 370364 2860 R 44.4 19.8 0:15.47 mem2.bash
0:16.47 1484 user 20 0 613208 393860 2860 R 44.4 21.1 0:16.47 mem2.bash
0:17.43 1484 user 20 0 634988 415508 2860 R 41.2 22.2 0:17.43 mem2.bash
0:18.42 1484 user 20 0 658220 438740 2860 R 41.2 23.5 0:18.42 mem2.bash
0:19.41 1484 user 20 0 680396 460916 2860 R 38.9 24.6 0:19.41 mem2.bash
0:20.39 1484 user 20 0 701912 482300 2860 R 35.3 25.8 0:20.39 mem2.bash
0:21.39 1484 user 20 0 724484 505004 2860 R 42.1 27.0 0:21.39 mem2.bash
0:22.37 1484 user 20 0 747452 527972 2860 R 47.1 28.2 0:22.37 mem2.bash
0:23.37 1484 user 20 0 771080 551468 2860 R 47.1 29.5 0:23.37 mem2.bash
0:24.36 1484 user 20 0 794840 575228 2860 R 38.9 30.7 0:24.36 mem2.bash
0:25.35 1484 user 20 0 818336 598724 2860 R 50.0 32.0 0:25.35 mem2.bash
0:26.33 1484 user 20 0 841568 622220 2860 R 41.2 33.3 0:26.33 mem2.bash
0:27.33 1484 user 20 0 865856 646244 2860 R 41.2 34.5 0:27.33 mem2.bash
0:28.31 1484 user 20 0 889484 670004 2860 R 41.2 35.8 0:28.31 mem2.bash
0:29.31 1484 user 20 0 913640 694028 2860 R 38.9 37.1 0:29.31 mem2.bash
0:30.30 1484 user 20 0 937400 717788 2860 R 42.1 38.4 0:30.30 mem2.bash
[Read 73 lines]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Uncut Text ^T To Spell ^_ Go To Line M-E Redo

График изменения свободной физической памяти и памяти раздела подкачки относительно времени



Вывод: Во втором случае сначала происходит полностью аналогичная ситуации с первым случаем, однако аварийно завершается только одна из программ (на 42-43с). После чего происходит резкий рост свободной физической памяти и памяти раздела подкачки (можно увижеть этот момент на графиках с 43 по 45 секунды). Затем физическая память вновь линейно убвает, а память раздела подкачки пока остается не изменной. После, когда физическая память заканчивается, начинает также убывать память раздела подкачки, пока программа не завершитсяя аварийно по ее истечению

Эксперимент №2

Скрипт newmem.bash:

```
 GNU nano 2.9.8                                    newmem.bash

#!/bin/bash
echo "" > report.log
arr=()
iter=0
while true; do
        arr+=( 1 2 3 4 5 6 7 8 9 10 )
        let iter=iter+1
        if [[ "${#arr[@]}" -ge "$1" ]]; then
                exit 0
        fi
done
```

Скрипт запуска run_newmem.bash:

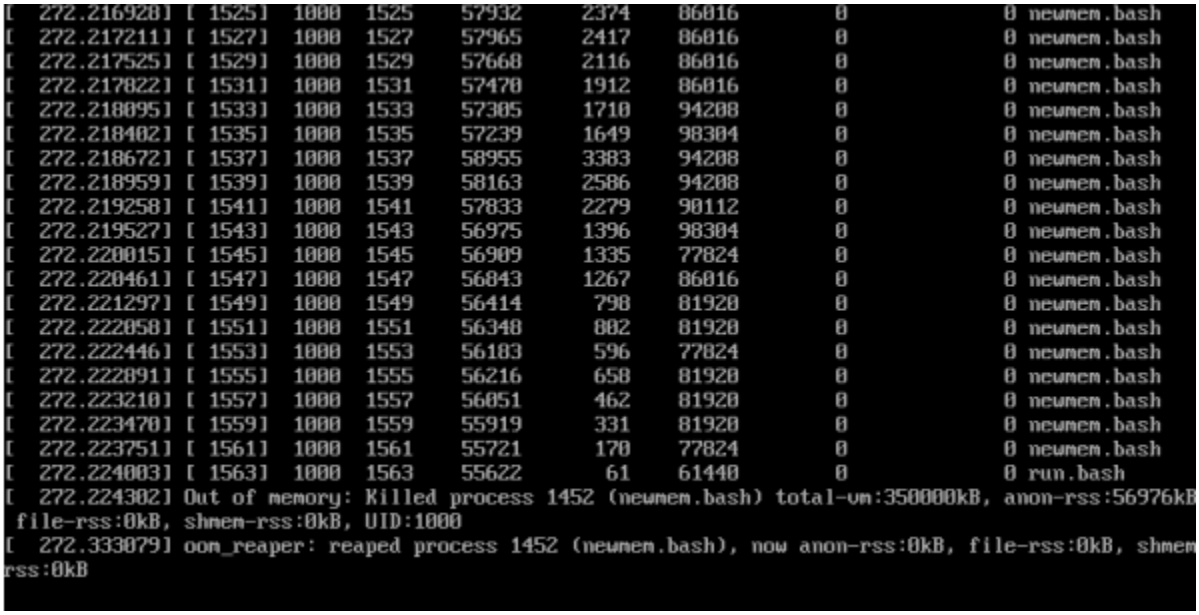
```
 GNU nano 2.9.8                                 run_newmem.bash

#!/bin/bash
for (( i=0; i < $2; i++ ))
do
        ./newmem.bash $1 $
        sleep 1
done
```

N(max) = 30 000 000 (из эксперимента 1)

При значении N в 10 раз меньшем, чем критическое (3 000 000) и при K=10, ни один из процессов не завершается аварийно:

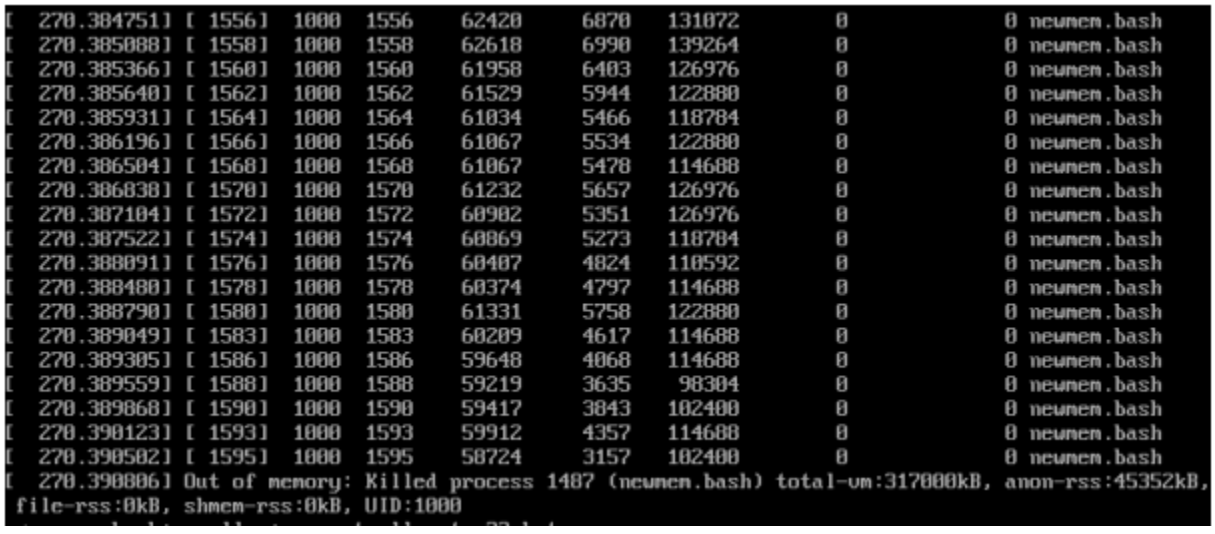
```
[user@localhost 2nd_experiment]$ ./run_newmem.bash 3000000 10
```

При запуске же N = 3 000 000 и K = 30, часть процессов завершилась аварийно. Процессор стал тратить больше времени на межстраничный обмен, чем на непосредственное выполнение программ.



При снижении до N = 2 000 000, K = 30 запусков успешно завершаются

При значении в N = 2 400 000 происходит аварийное завершение части программ.



Таким образом, максимальное значение N ≈ 2 300 000

```
[user@localhost 2nd_experiment]$ ./run_newmem.bash 2300000 30
[user@localhost 2nd_experiment]$
```