Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение
высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Лабораторная работа №5

по дисциплине «**Операционные системы**»

Управление памятью в ОС Linux

Автор: Юрпалов Сергей Николаевич

Группа: M3205

Преподаватель: Маятин Александр Владимирович

Санкт-Петербург, 2021

Данные о текущей конфигурации OS:

- Общий объём оперативной памяти: 2048 Мб
- Объём раздела подкачки: 820 Мб
- Размер страницы виртуальной памяти: 4 Кб
- Объём свободной физической памяти в ненагруженной системе: 1618.5 Мб
- Объём свободного пространства в разделе подкачки в ненагруженной системе: 820 Мб

Эксперимент №1

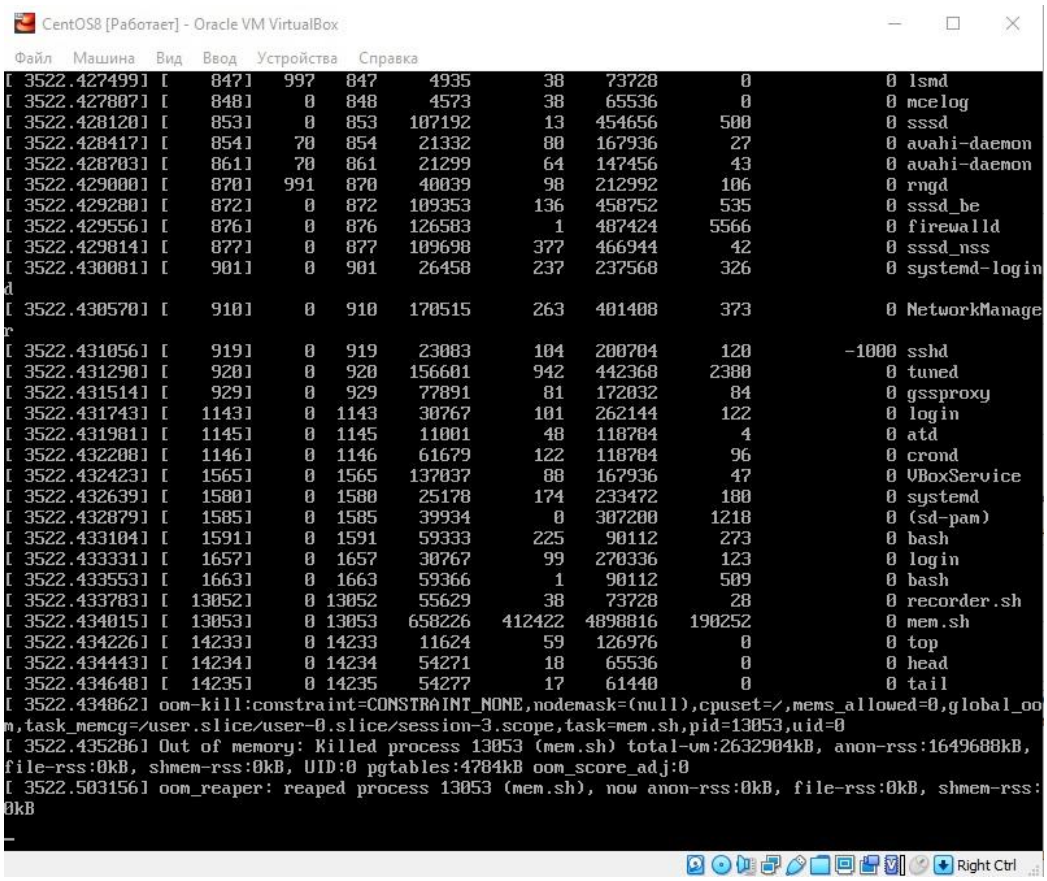
Подготовительный этап:

1. Написал скрипт mem.sh, который в бесконечном цикле добавляет в конец массива последовательность из 100 элементов (для ускорения процесса исследования) и каждую 10000 итерацию записывает его длину в файл report.log.

```
#!/bin/bash

report="/home/user/OS/lab5/exp1/report.log"
cnt=0

declare -a array
declare -a numbers=(1,2,3,4,5,6,7,8,9,10,11,

rm $report

while :
do
    array+=(${numbers[@]})
    cnt=$((cnt+1))
    if [[ $cnt -eq 10000 ]]
    then
        echo "${#array[@]}" >> $report
        cnt=0
    fi
done
```

Первый этап:

1. Последняя запись журнала – значения параметров, с которыми произошла аварийная остановка.



2. Значение в последней строке report.log – 7 010 000

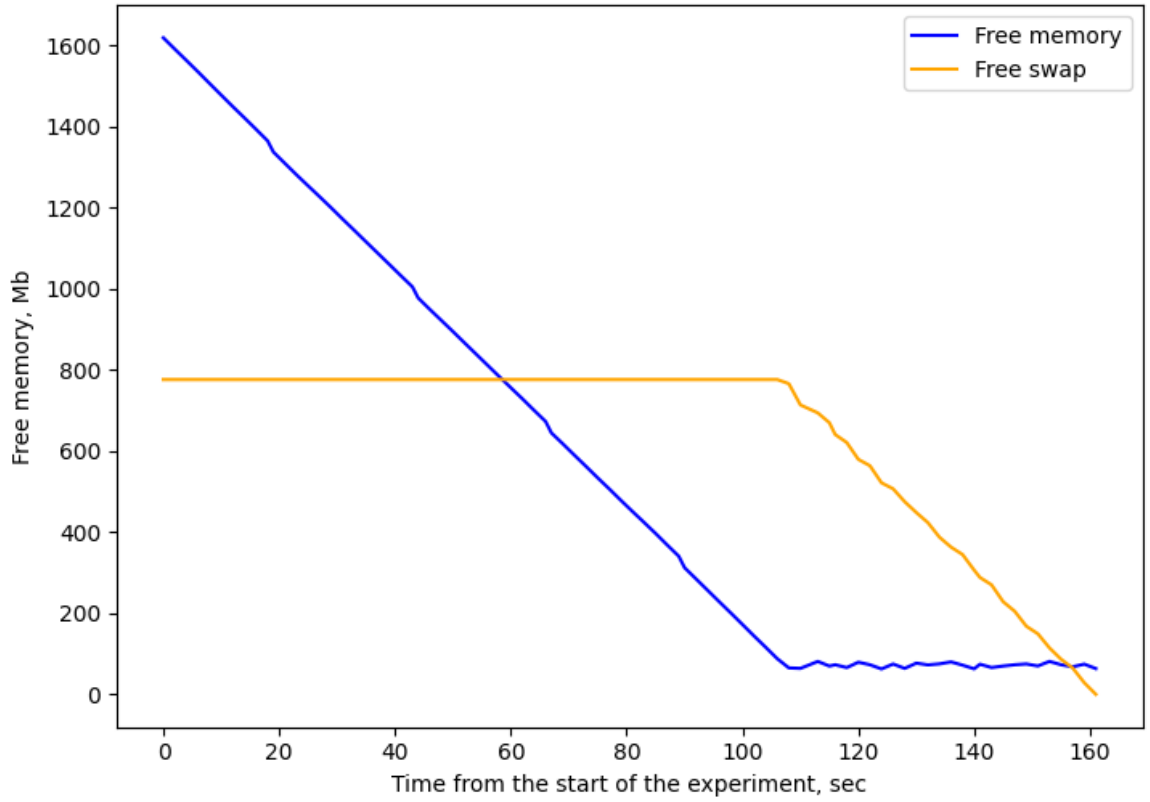
3. Данные, полученные в процессе исполнения mem.sh записаны в файл parameters.log

```
12:51:48

System parameters:
MiB Mem :   1817.3 total,   1618.5 free,    136.0 used,     62.9 buff/cache
MiB Swap:    820.0 total,    776.0 free,     44.0 used.   1573.0 avail Mem

mem.sh parameters:
  13053 root      20   0  230708  11348   2964 R  99.9   0.6   0:00.55 mem.sh

Top 5 processes parameters:
  13053 root      20   0  234164  14804   2964 R  88.2   0.8   0:00.78 mem.sh
      1 root      20   0  186376   2780      0 S   0.0   0.1   0:00.74 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
12:51:50

System parameters:
MiB Mem :   1817.3 total,   1590.3 free,    164.1 used,     62.9 buff/cache
MiB Swap:    820.0 total,    776.0 free,     44.0 used.   1544.8 avail Mem

mem.sh parameters:
  13053 root      20   0  259512  40024   2964 R  99.9   2.2   0:02.42 mem.sh

Top 5 processes parameters:
  13053 root      20   0  262840  43480   2964 R  93.8   2.3   0:02.65 mem.sh
      1 root      20   0  186376   2780      0 S   0.0   0.1   0:00.74 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

4. Записи о скрипте в системном журнале

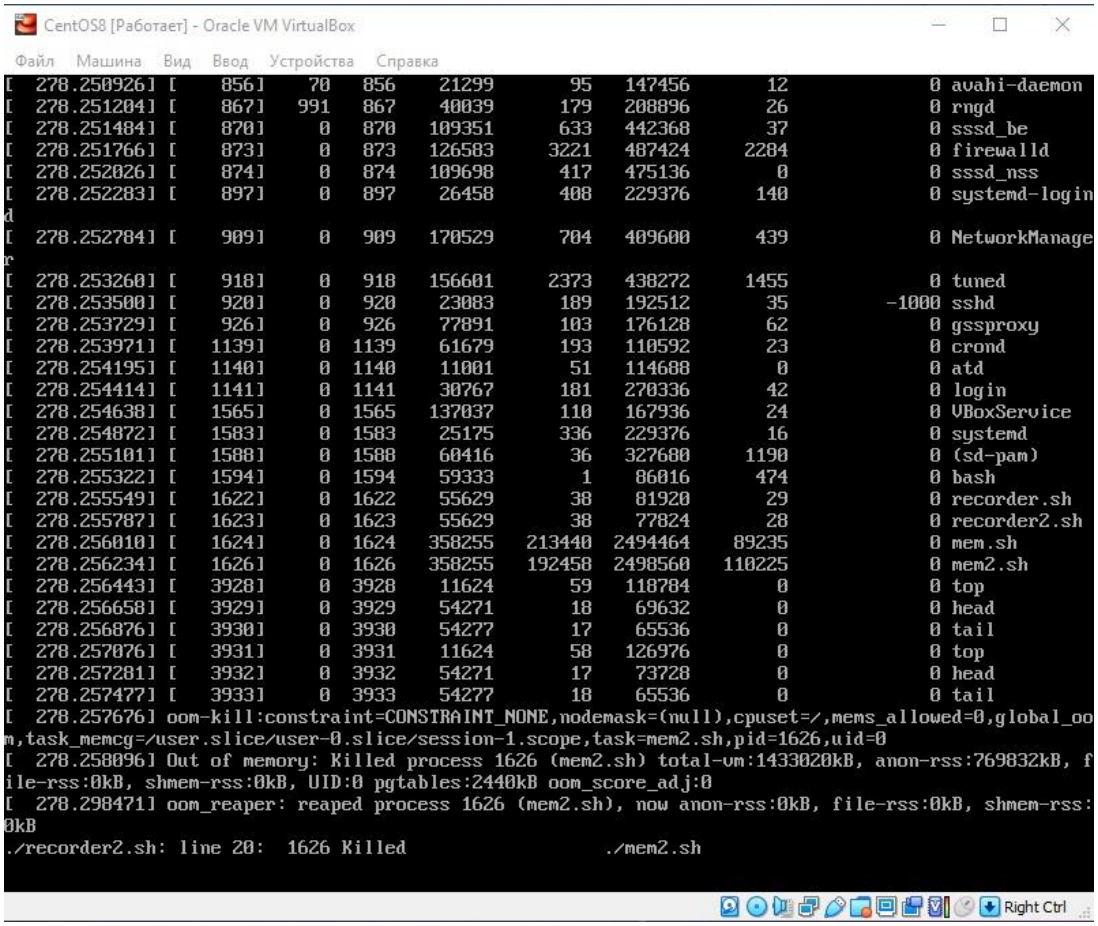
```
[ 2931.421389] [   9132]     0  9132   660754   406729  4935680   198457             0 mem.sh
[ 2931.422443] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,
mems_allowed=0,global_oom,task_memcg=/user.slice/user-0.slice/session-3.scope,task=mem.sh,pid=9132,uid=0
[ 2931.422866] Out of memory: Killed process 9132 (mem.sh) total-vm:2643016kB,
 anon-rss:1626916kB, file-rss:0kB, shmem-rss:0kB, UID:0 pgtables:4820kB oom_score_adj:0
[ 2931.491438] oom_reaper: reaped process 9132 (mem.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
[ 3522.434015] [  13053]     0 13053   658226   412422  4898816   190252             0 mem.sh
[ 3522.434862] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,
global_oom,task_memcg=/user.slice/user-0.slice/session-3.scope,task=mem.sh,pid=13053,uid=0
[ 3522.435286] Out of memory: Killed process 13053 (mem.sh) total-vm:2632904kB,
anon-rss:1649688kB, file-rss:0kB, shmem-rss:0kB, UID:0 pgtables:4784kB oom_score_adj:0
[ 3522.503156] oom_reaper: reaped process 13053 (mem.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
```

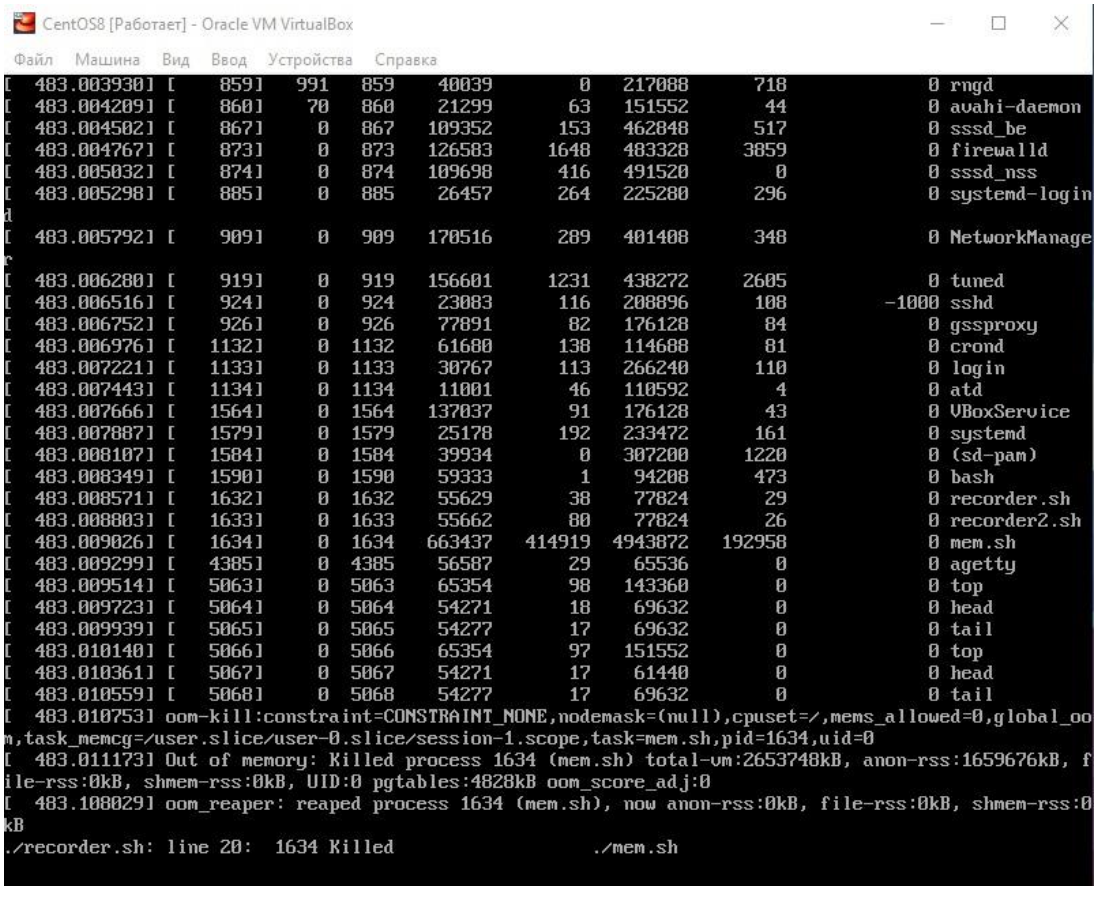
5. График зависимости свободной памяти от времени по данным п3, построен с помощью matplotlib python 3.



Второй этап:

1. Последняя запись журнала – значения параметров, с которыми произошла аварийная остановка.





2. Значение в последней строке report.log – 7 000 000, report2.log – 3 520 000.
3. Данные, полученные в процессе исполнения mem.sh записаны в файл parameters.log, для mem2.sh аналогично в parameters2.log

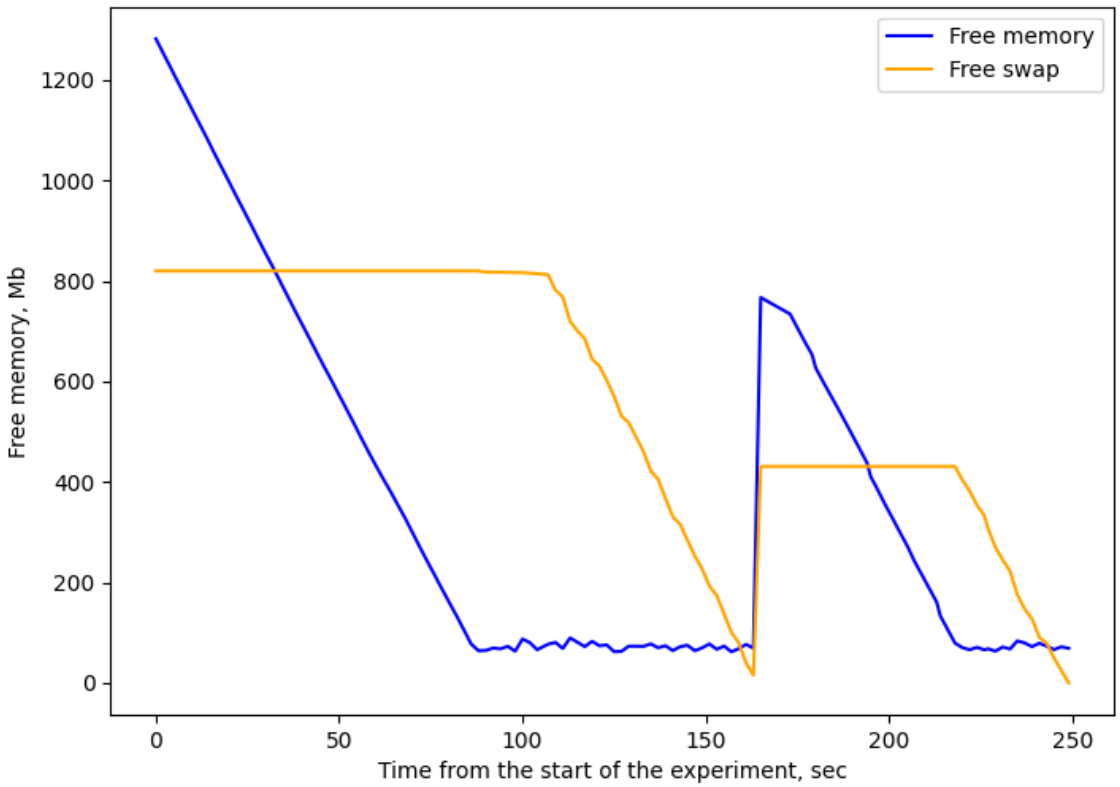
```
14:25:54

System parameters:
MiB Mem :   1817.3 total,   1282.1 free,    167.6 used,    367.6 buff/cache
MiB Swap:    820.0 total,    820.0 free,      0.0 used.   1490.1 avail Mem

mem.sh parameters:
   1624 root      20   0  227636   8084   2904 R  41.2   0.4   0:00.32 mem.sh

Top 5 processes parameters:
   1624 root      20   0  229428  10004   2904 R  43.8   0.5   0:00.44 mem.sh
   1626 root      20   0  229300   9856   2884 R  43.8   0.5   0:00.43 mem2.sh
      1 root      20   0  186376  14336   9672 S   0.0   0.8   0:00.71 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp

14:25:54

System parameters:
MiB Mem :   1817.3 total,   1282.7 free,    167.0 used,    367.6 buff/cache
MiB Swap:    820.0 total,    820.0 free,      0.0 used.   1489.3 avail Mem

mem2.sh parameters:
   1626 root      20   0  227508   7936   2884 R  52.9   0.4   0:00.32 mem2.sh

Top 5 processes parameters:
   1624 root      20   0  229300   9876   2904 R  43.8   0.5   0:00.43 mem.sh
   1626 root      20   0  229044   9600   2884 R  43.8   0.5   0:00.42 mem2.sh
      1 root      20   0  186376  14336   9672 S   0.0   0.8   0:00.71 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
```

4. Записи о скрипте в системном журнале

```
[  278.256010] [   1624]     0  1624   358255   213440  2494464    89235             0 mem.sh
[  380.083131] [   1624]     0  1624   657842   409027  4894720   193240             0 mem.sh
[  380.084615] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,
global_oom,task_memcg=/user.slice/user-0.slice/session-1.scope,task=mem.sh,pid=1624,uid=0
[  380.085056] Out of memory: Killed process 1624 (mem.sh) total-vm:2631368kB,
 anon-rss:1636108kB, file-rss:0kB, shmem-rss:0kB, UID:0 pgtables:4780kB oom_score_adj:0
[  380.159956] oom_reaper: reaped process 1624 (mem.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
```

```
[  278.256234] [   1626]     0  1626   358255   192458  2498560   110225             0 mem2.sh
[  278.257676] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,
global_oom,task_memcg=/user.slice/user-0.slice/session-1.scope,task=mem2.sh,pid=1626,uid=0
[  278.258096] Out of memory: Killed process 1626 (mem2.sh) total-vm:1433020kB, anon-rss:769832kB,
 file-rss:0kB, shmem-rss:0kB, UID:0 pgtables:2440kB oom_score_adj:0
[  278.298471] oom_reaper: reaped process 1626 (mem2.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
```

5. График зависимости свободной памяти от времени по данным п3, построен с помощью matplotlib python 3.



Выводы по эксперименту №1:

- В случае запуска одного скрипта, при малых значениях свободной физической памяти (в моём случае <60 Мб) система переходит на использование файлов подкачки. В момент, когда их свободное значение падает до 0, происходит аварийное завершение процесса.
- В случае запуска двух скриптов сразу, в начале на графике можно увидеть картину, схожую с п1. Далее, в момент ~160 секунд от запуска скриптов, mem2.sh аварийно завершает свою работу, освобождая как физическую память, так и файлы подкачки. После чего ситуация для mem.sh аналогична п1.

Эксперимент №2

Подготовительный этап:

1. Изменён скрипт mem.sh, который теперь завершает своё исполнение при определённом размере массива.

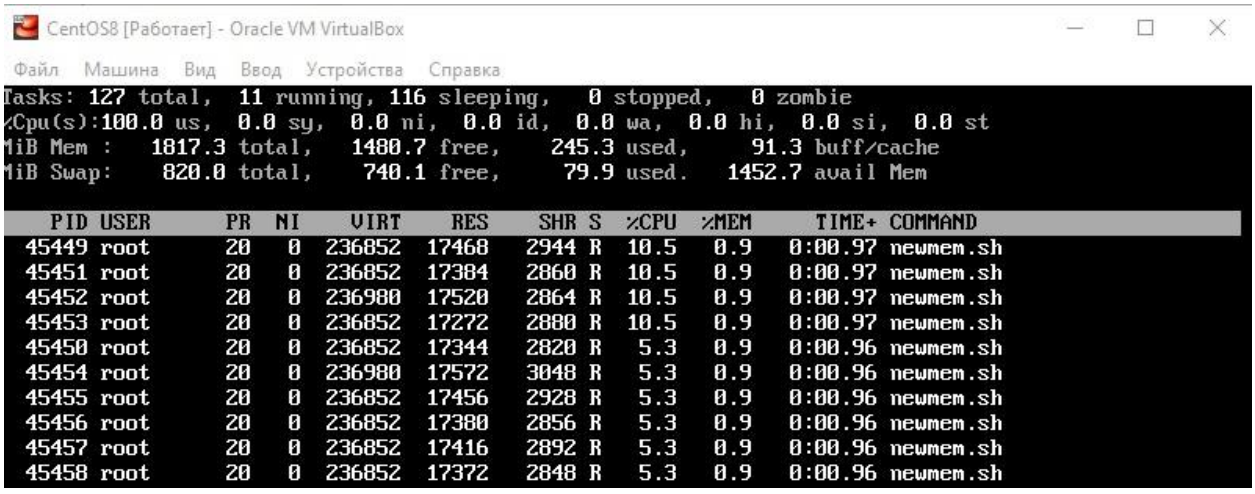
```
#!/bin/bash

report="/home/user/OS/lab5/exp2/report.log"
N=$1

declare -a array
declare -a numbers=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,

#rm $report

while :
do
    array+=(${numbers[@]})
    if [[ "${#array[@]}" -gt "$N" ]]
    then
        echo "Successfully finished with array size - $N" >> $report
        exit 0
    fi

done
```

Основной этап:

1. Написан скрипт для запуска newmem.sh в количестве K с заданным N.

```
#!/bin/bash

K=$1
N=$2

for i in $(seq 1 $K)
do
    ./newmem.sh $N&
done
```

2. Произведён запуск 10 newmem.sh с заданным N = 7 000 000 / 10 = 700 000, все процессы завершились успешно.



report.log – Блокнот

Файл Правка Формат Вид Справка

```
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
Successfully finished with array size - 700000
```

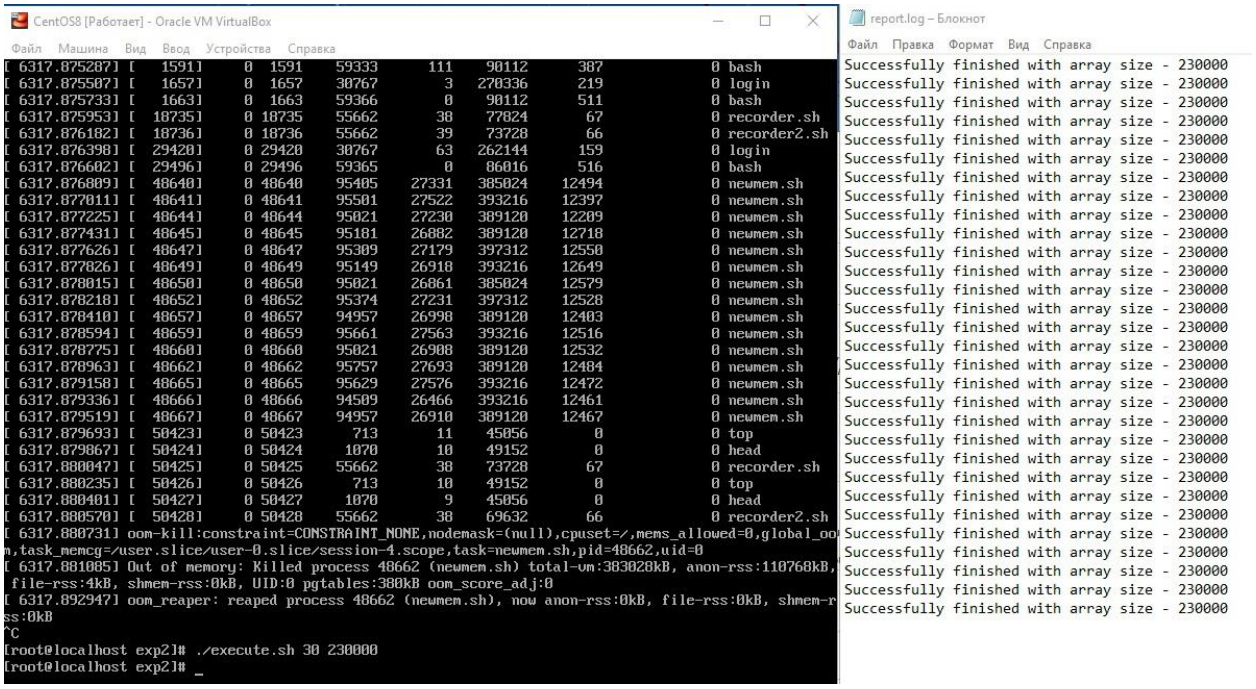
3. Произведён запуск 30 newmem.sh с заданным N = 700 000, процессы после 10 были завершены аварийно с соответствующим сообщением, ниже приведён пример одного из них.

CentOS8 [Работает] - Oracle VM VirtualBox

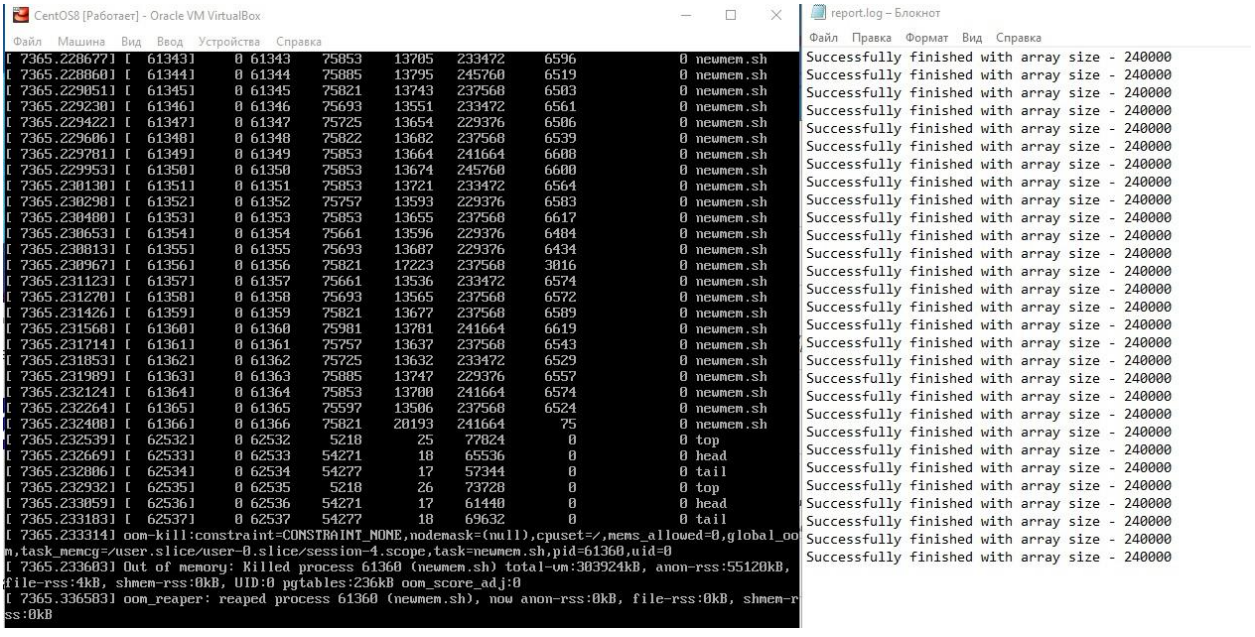
Файл Машина Вид Ввод Устройства Справка

```
[ 6363.871648] [  29496]     0 29496    59365        0    86016      516             0 bash
[ 6363.871854] [  48640]     0 48640   105517    34436   466944    15501             0 newmem.sh
[ 6363.872068] [  48644]     0 48644   105101    33256   471040    16264             0 newmem.sh
[ 6363.872268] [  48645]     0 48645   104781    33588   462848    15612             0 newmem.sh
[ 6363.872473] [  48647]     0 48647   104845    33710   475136    15555             0 newmem.sh
[ 6363.872667] [  48649]     0 48649   109773    38584   512000    15608             0 newmem.sh
[ 6363.872857] [  48650]     0 48650   104493    33455   462848    15457             0 newmem.sh
[ 6363.873075] [  48652]     0 48652   109901    38984   512000    15336             0 newmem.sh
[ 6363.873264] [  48657]     0 48657   105005    34064   466944    15362             0 newmem.sh
[ 6363.873456] [  48660]     0 48660   105197    34164   471040    15452             0 newmem.sh
[ 6363.873641] [  48665]     0 48665   105069    34074   471040    15414             0 newmem.sh
[ 6363.873821] [  48666]     0 48666   108781    37862   507904    15338             0 newmem.sh
[ 6363.874024] [  48667]     0 48667   104461    33507   466944    15374             0 newmem.sh
[ 6363.874207] [  50692]     0 50692     6277       29    81920        0             0 top
[ 6363.874387] [  50693]     0 50693    54271       17    65536        0             0 head
[ 6363.874573] [  50694]     0 50694    54277       17    65536        0             0 tail
[ 6363.874750] [  50695]     0 50695     6277       28    86016        0             0 top
[ 6363.874924] [  50696]     0 50696    54271       17    65536        0             0 head
[ 6363.875118] [  50697]     0 50697    54277       17    61440        0             0 tail
[ 6363.875286] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oo
m,task_memcg=/user.slice/user-0.slice/session-4.scope,task=newmem.sh,pid=48652,uid=0
[ 6363.875643] Out of memory: Killed process 48652 (newmem.sh) total-vm:439604kB, anon-rss:155932kB,
 file-rss:4kB, shmem-rss:0kB, UID:0 pgtables:500kB oom_score_adj:0
[ 6363.889162] oom_reaper: reaped process 48652 (newmem.sh), now anon-rss:0kB, file-rss:0kB, shmem-r
ss:0kB
[ 6380.010855] newmem.sh invoked oom-killer: gfp_mask=0x6200ca(GFP_HIGHUSER_MOVABLE), order=0, oom_s
core_adj=0
[ 6380.011314] CPU: 0 PID: 48645 Comm: newmem.sh Kdump: loaded Tainted: G           OE    ---------r
-  - 4.18.0-305.25.1.el8_4.x86_64 #1
[ 6380.011771] Hardware name: innotek GmbH VirtualBox/VirtualBox, BIOS VirtualBox 12/01/2006
```

4. Вычислим такое значение N, что при K = 30, все скрипты будут завершены успешно. Логично, что для этого необходимо 7 000 000 / 30 = 230 000 с округлением в меньшую сторону. Проверим запуском при K = 30, N = 230 000. Наши расчёты оказались верны.



5. Проверим, что мы действительно нашли максимальное значение – сделаем тестовый запуск при K = 30, N = 240 000. Действительно, последний скрипт newmem.sh завершается аварийно с соответствующей ошибкой. В журнале лишь 29 записей об успешном завершении.



Вывод по эксперименту №2:

Найденное экспериментально значение N = 230 000 = 7 000 000 / 30.

Вывод:

В ходе работы я на практике убедился, что во время работы с оперативной памятью в системе Linux данные заполняют физическую память, пока она не дойдёт до критически низкого значения. После этого, OS начинает заполнять файлы подкачки. Если и этого оказалось недостаточно, происходит аварийная остановка процесса.