МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное
учреждение высшего образования Санкт-Петербургский национальный
исследовательский университет информационных технологий
механики и оптики
Мегафакультет трансляционных информационных технологий
Факультет информационных технологий и программирования

Лабораторная работа № 5 по дисциплине «Операционные системы» Управление памятью в ОС Linux

Выполнила студентка группы №M32041
Петренко Людмила Евгеньевна
Преподаватель: Хегай Максим Вилорьевич

САНКТ-ПЕТЕРБУРГ
2020

Данные о текущей конфигурации ОС - получены командой free -m:

- Общий объем оперативной памяти: 2048 Мб
- Объем раздела подкачки: 820 Мб
- Размер страницы виртуальной памяти: 4 Кб
- Объем свободной физической памяти в ненагруженной системе: 1498 Мб
- Объем свободного пространства в разделе подкачки в ненагруженной системе: 820 Мб

Первый эксперимент.
Подготовительный этап:

Создан скрипт mem.bash, который выполняет бесконечный цикл. Перед началом выполнения цикла создается пустой массив и счетчик шагов, инициализированный нулем. На каждом шаге цикла в конец массива добавляется последовательность из 10 элементов, (1 2 3 4 5 6 7 8 9 10). Каждый 100000-ый шаг в файл report.log добавляется строка с текущим значением размера массива (перед запуском скрипта, файл обнуляется)

```
[user@localhost lab5]$ cat mem.sh
#!/bin/bash

array=()
count=0
>report.log
while [ true ]
do
        array+=(1 2 3 4 5 6 7 8 9 10)
        let "count += 1"
        if (( $count % 100000 == 0 ))
        then
                echo ${#array[@]} >> report.log
        fi
done
```

Первый этап:

1. Запустить mem.sh, дождаться аварийной остановки и зафиксировать значения.
   Последняя запись журнала - значения параметров, с которыми произошла аварийная остановка процесса.

```
[ 9541.672602] [ 1842]  1000  1842    82055        1   290816    26485             0 mem.sh
[ 9541.677702] [28087]  1000 28087   634739   419817  4710400   159357             0 mem.sh
[ 9541.679286] Out of memory: Killed process 28087 (mem.sh) total-vm:2538956kB, anon-rss:1679268kB,
file-rss:0kB, shmem-rss:0kB, UID:1000
[ 9541.828776] oom_reaper: reaped process 28087 (mem.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:
0kB
```

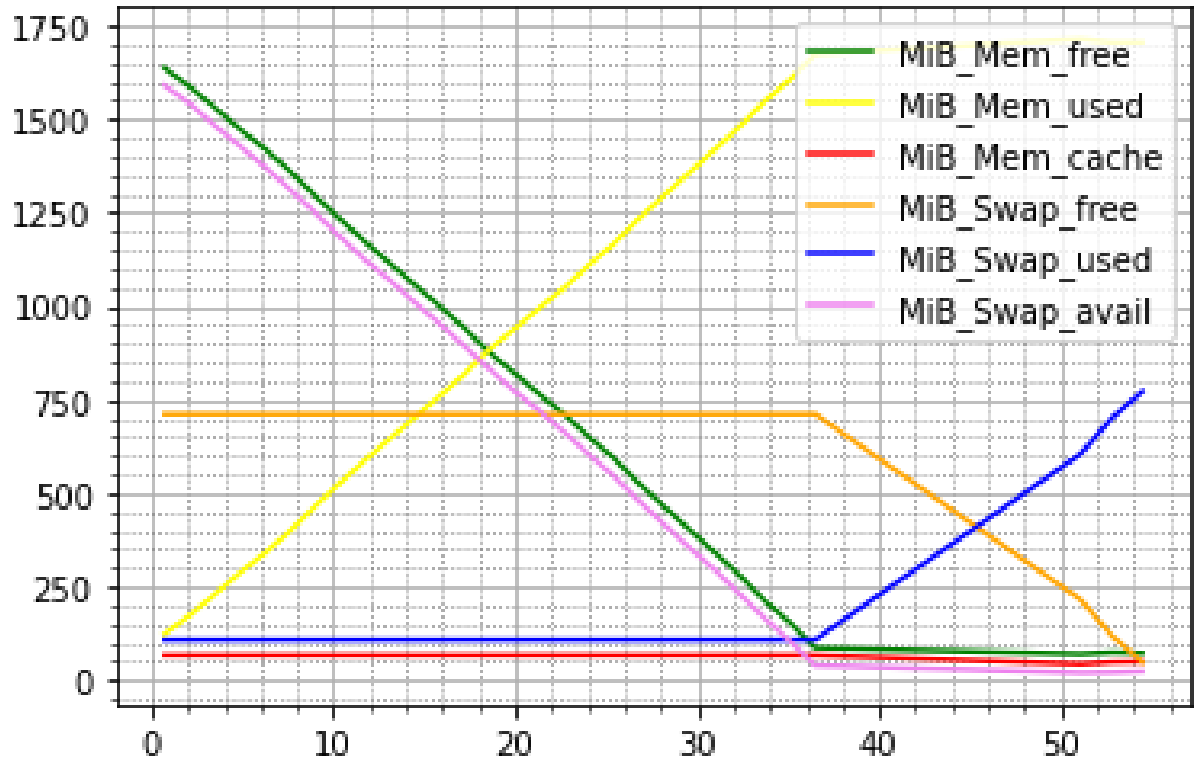
Последнее значение в report.log = 29000000.

2. Запустить mem.sh, отслеживая значения параметров с помощью команды top.
   Последняя запись журнала:

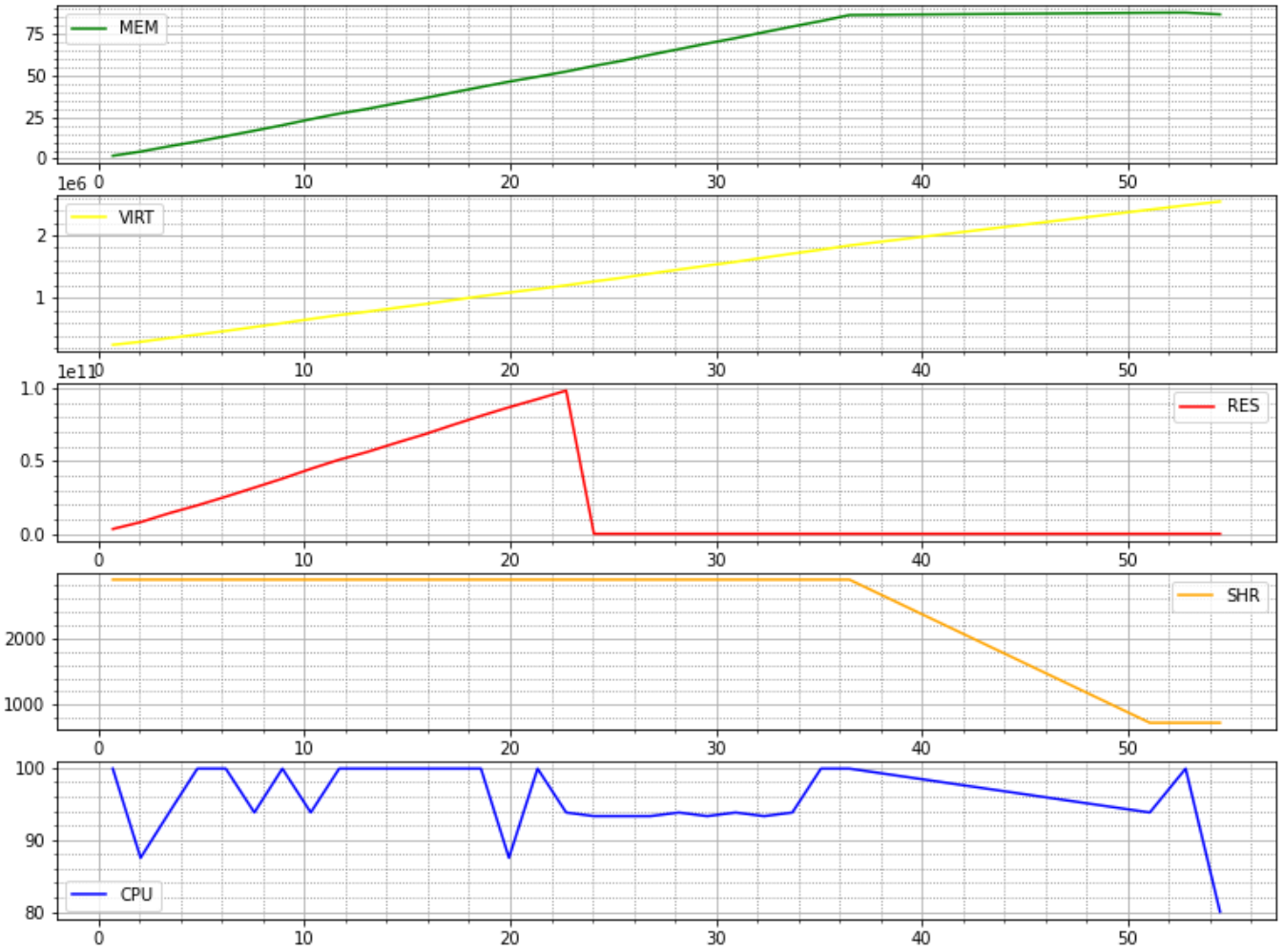
```
[ 9541.672602] [ 1842]  1000  1842    82055        1   290816    26485             0 mem.sh
[ 9541.677702] [28087]  1000 28087   634739   419817  4710400   159357             0 mem.sh
[ 9541.679286] Out of memory: Killed process 28087 (mem.sh) total-vm:2538956kB, anon-rss:1679268kB,
file-rss:0kB, shmem-rss:0kB, UID:1000
[ 9541.828776] oom_reaper: reaped process 28087 (mem.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:
0kB

[21939.507195] [33187]  1000 33187   657740   418720  4902912   183458             0 mem.sh
[21939.508774] Out of memory: Killed process 33187 (mem.sh) total-vm:2630960kB, anon-rss:1674880kB,
file-rss:0kB, shmem-rss:0kB, UID:1000
[21939.633034] oom_reaper: reaped process 33187 (mem.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:
0kB
```

Последнее значение в report.log = 30000000.
Представленные в графическом формате ниже данные: ссылка

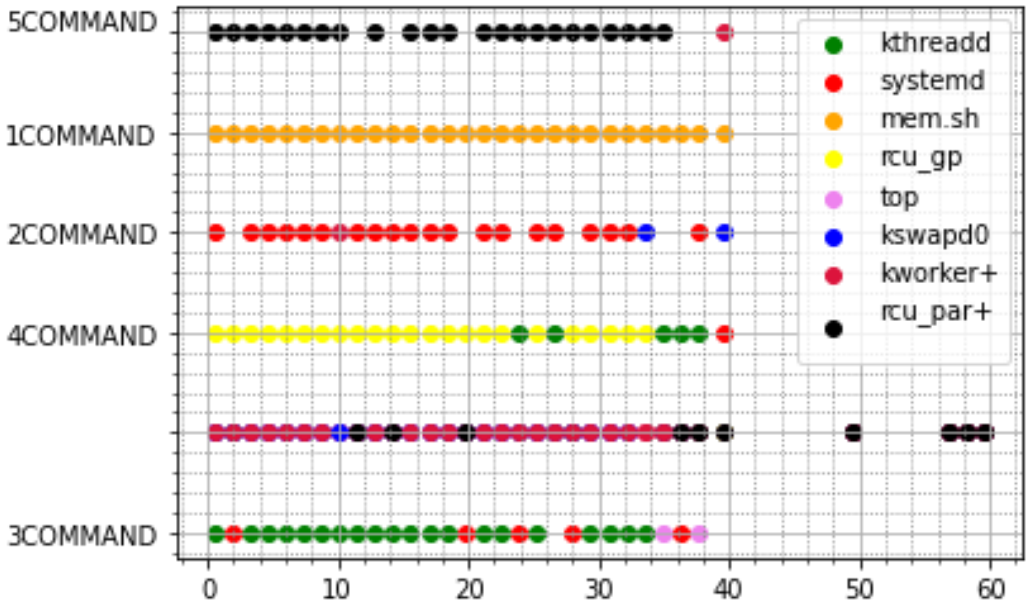
Динамика значений параметров памяти системы:



Динамика значений параметров работающего скрипта:



Состав и позиции первых пяти процессов:



Выводы:

По графикам видно, что при низкой величине физической свободной памяти, начинают использоваться файлы подкачки. Когда их тоже не хватает - происходит аварийное завершение программы.

Второй этап.

1. Сделать копию скрипта mem.sh : mem2.sh и отслеживать изменения параметров при работе нескольких экземпляров скрипта.

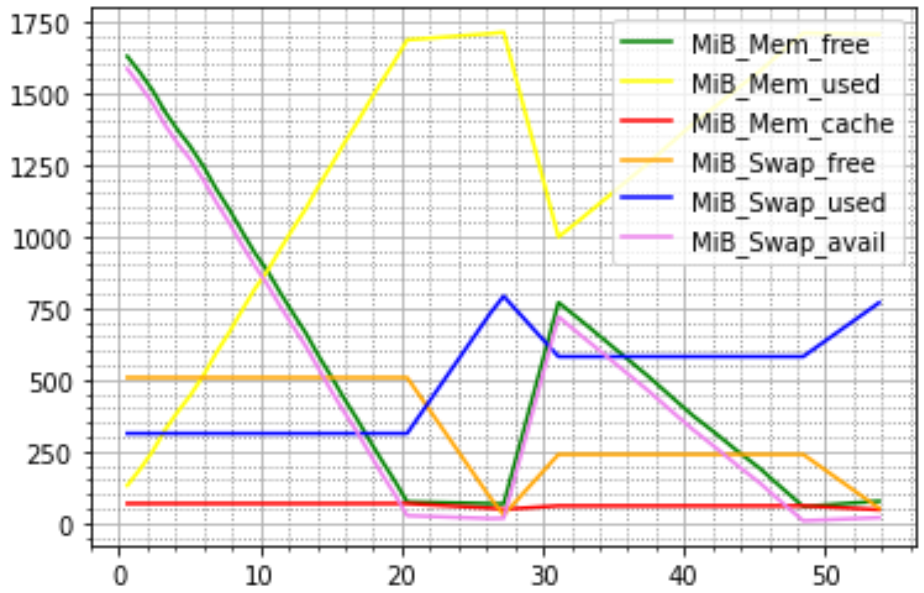
Последняя запись журнала:

```
[43446.523925] mem.sh invoked oom-killer: gfp_mask=0x6200ca(GFP_HIGHUSER_MOVABLE), no
demask=(null), order=0, oom_score_adj=0
[43446.526828] mem.sh cpuset=/ mems_allowed=0
[43446.528170] CPU: 0 PID: 40018 Comm: mem.sh Kdump: loaded Tainted: G
  ---------r-  - 4.18.0-193.el8.x86_64 #1
[43446.670001] [35970]  1000 35970    81098        1   278528    25554             0
mem.sh
[43446.671819] [36070]  1000 36070    68162        1   172032    12587             0
mem.sh
[43446.672109] [36072]  1000 36072    68723        1   192512    13175             0
mem2.sh
[43446.678627] [40018]  1000 40018   602267   417783  4456448   129013             0
mem.sh
[43446.680139] Out of memory: Killed process 40018 (mem.sh) total-vm:2409068kB, anon-
rss:1670668kB, file-rss:464kB, shmem-rss:0kB, UID:1000
[43446.894625] oom_reaper: reaped process 40018 (mem.sh), now anon-rss:0kB, file-rss:
0kB, shmem-rss:0kB
```

Последнее значение в report.log = 27000000.
Последнее значение в report2.log = 14000000.

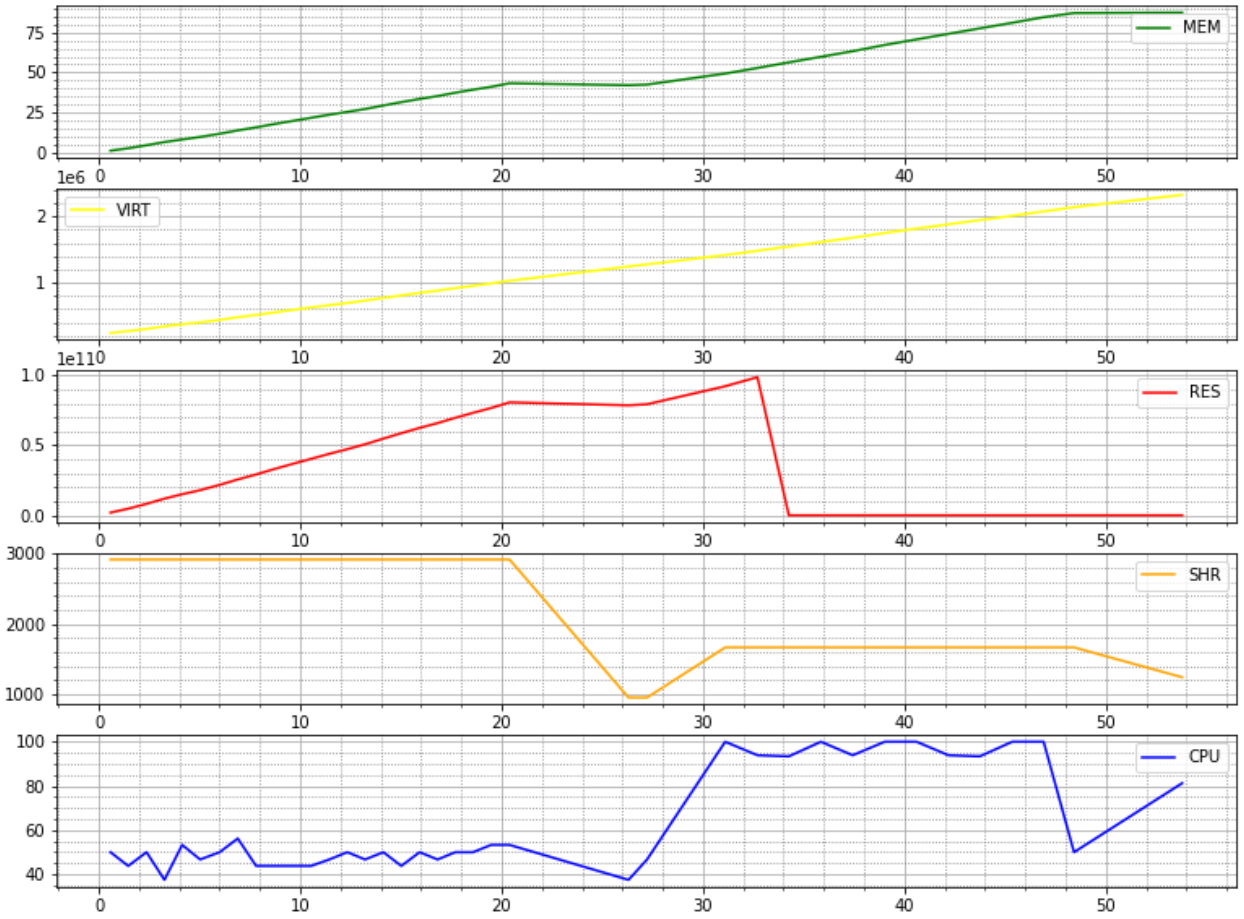
Представленные в графическом формате ниже данные: ссылка

Динамика значений параметров памяти системы:

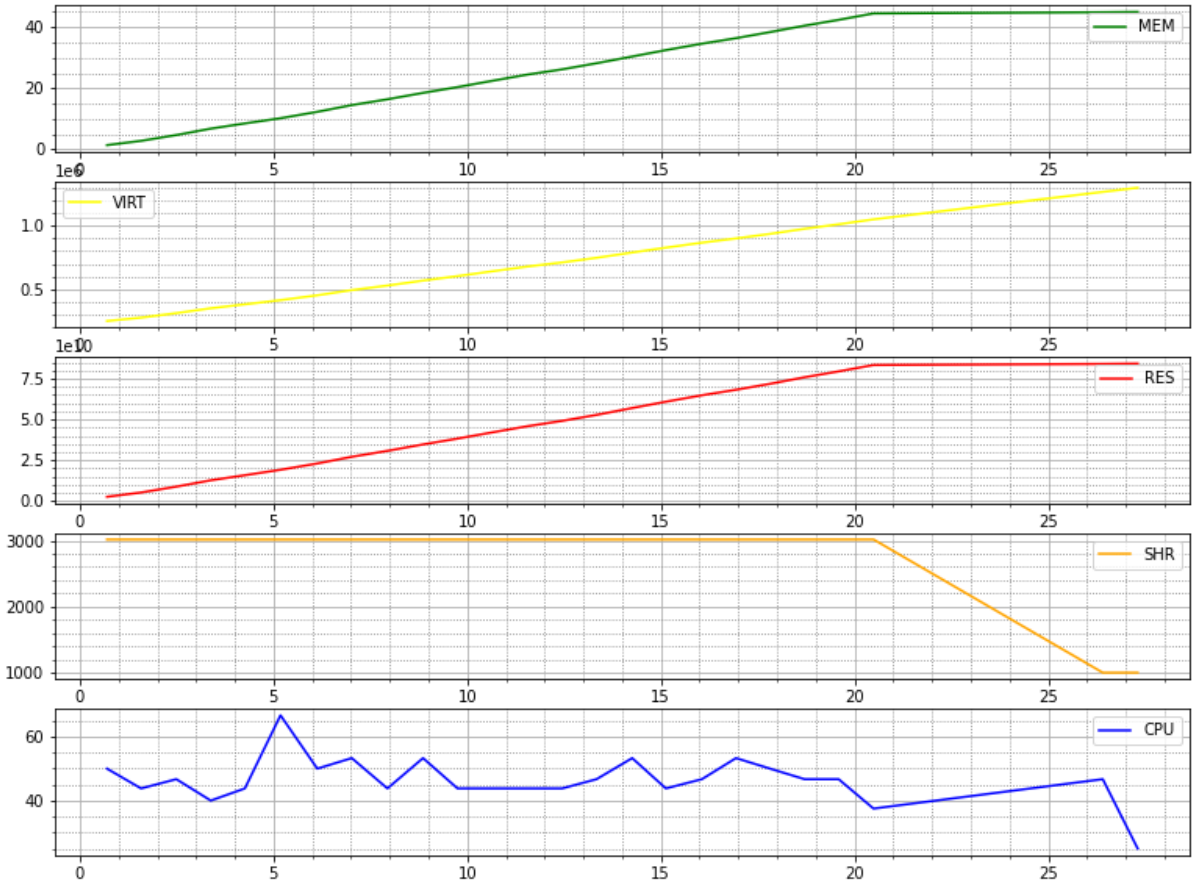


Динамика значений параметров работающего скрипта:

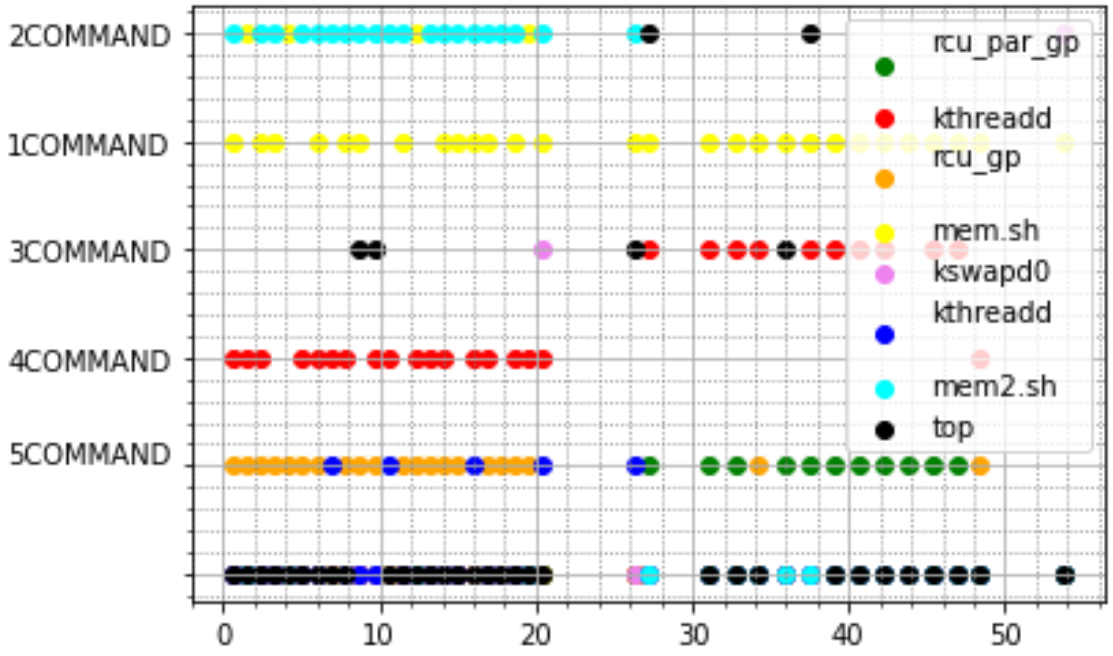
1. mem.sh



2. mem2.sh



Состав и позиции первых пяти процессов:



Выводы:

По графику видно, что если запустить два скрипта одновременно, один из них завершится раньше, а второй продолжит выполняться как если бы он изначально был один. Это связано с тем, что после завершения первого скрипта восстанавливается физическая память, которую он занимал.

Второй эксперимент.
Подготовительный этап:
Создать копию скрипта mem.bash в файл newmem.bash, сделав так, чтобы она завершала работу, как только размер создаваемого массива превысит значение N, передаваемое в качестве параметра скрипту. Убрать запись данных в файл.

```
#!/bin/bash
array=()
counter=0
N=$1
while [ true ]
do
    array+=(1 2 3 4 5 6 7 8 9 10)

    if [ ${#array[@]} -gt $N ]
    then
        exit 1
    fi
done
```

Основной этап:
Определить граничные значения потребления памяти, обеспечивающие безаварийную работу для регулярных процессов, запускающихся с заданной интенсивностью.

1. Создать скрипт, который будет запускать newmem.bash каждую секунду, используя один и тот же параметр N так, что всего будет осуществлено K запусков.

   N - величина, в 10 раз меньшую, чем размер массива, при котором происходила аварийная остановка процесса в первом этапе предыдущего эксперимента.
   K = 10
2. Убедиться, что все K запусков успешно завершились, и в системном журнале нет записей об аварийной остановке newmem.bash. (N = 300000)

```
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 156622 user      20   0  237668  18316   3000 R  10.3   1.0   0:00.44 newmem.sh
 156620 user      20   0  238592  18988   2876 R   9.9   1.0   0:00.44 newmem.sh
 156621 user      20   0  238328  19084   2980 R   9.9   1.0   0:00.44 newmem.sh
 156625 user      20   0  237404  18028   2972 R   9.9   1.0   0:00.44 newmem.sh
 156629 user      20   0  237404  18048   3000 R   9.9   1.0   0:00.44 newmem.sh
 156626 user      20   0  238460  19112   3008 R   9.6   1.0   0:00.43 newmem.sh
 156627 user      20   0  238196  18856   3016 R   9.6   1.0   0:00.43 newmem.sh
 156623 user      20   0  237668  18312   2996 R   9.3   1.0   0:00.42 newmem.sh
 156624 user      20   0  236744  17244   2988 R   9.3   0.9   0:00.42 newmem.sh
 156628 user      20   0  237272  18048   3000 R   9.3   1.0   0:00.43 newmem.sh
 156541 user      20   0  274536   1884   1244 R   0.3   0.1   0:07.08 top
```

Программа успешно завершилась.

3. Изменить значение K на 30 и снова запустить скрипт.
   При K = 30 и N = 900000 - программа успешно завершилась.
   При K = 30 и N = 1100000 - программа завершилась аварийно.
   При K = 30 и N = 1000000 - программа успешно завершилась.
   При K = 30 и N = 1050000 - программа завершилась аварийно.

4. Подобрать такое максимальное значение N, чтобы при K=30 не происходило аварийных завершений процессов.

Выводы:
Найденное значение N = 1000000.