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

Отчет по лабораторной работе №5

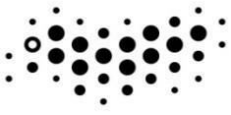
«Управление памятью в ОС Linux»

Автор: Бессонницын Евгений Сергеевич

Факультет: ФИТиП

Группа: М3234

Преподаватель: Маятин Александр Владимирович



УНИВЕРСИТЕТ ИТМО

Санкт-Петербург 2021

Рассматриваемые вопросы:

1. Использование утилиты top для мониторинга параметров памяти
2. Использование имитационных экспериментов для анализа работы механизмов управления памятью.

Основные источники данных о состоянии памяти вычислительного узла:

- команда ***free***
- файл ***/proc/meminfo*** (документация в соответствующем разделе man proc)
- файл /***proc/[PID]/statm*** (документация в соответствующем разделе man proc
- утилита ***top***

Организация управления памятью в ОС Linux:

В Linux используется страничная организация виртуальной памяти. Память разбита на страницы. Размер страницы можно посмотреть в параметрах конфигурации с помощью команды ***getconf PAGE_SIZE***. При обращении к адресу в памяти происходит динамическое преобразование адреса путем замены старших бит виртуального адреса на номер физической страницы с сохранением значения младших бит как смещения на странице.

Обычно, кроме физической памяти используется также раздел подкачки. В этом случае адресное пространство процесса состоит из страниц, находящихся в оперативной памяти и страниц, находящихся в разделе подкачки. Параметры раздела подкачки можно узнать из файла ***/proc/swaps***. При динамическом выделении памяти процессу, операционная система сначала пытается выделить страницы в физической памяти, но если это невозможно, инициирует страничный обмен, в рамках которого ряд страниц из физической памяти вытесняется на раздел подкачки, а адреса, соответствующие вытесненным страницам выделяются процессу под новые страницы.

Операционная система контролирует выделение памяти процессам. Если процесс попробует запросить расширение адресного пространства, которое невозможно в пределах имеющейся свободной оперативной памяти, его работа будет аварийно остановлена с записью в системном журнале.

Задание на лабораторную работу:

Проведите два виртуальных эксперимента в соответствии с требованиями и проанализируйте их результаты. В указаниях ниже описано, какие данные необходимо фиксировать в процессе проведения экспериментов. Рекомендуется написать «следящие» скрипты и собирать данные, например, из вывода утилиты top автоматически с заданной периодичностью, например, 1 раз в секунду. Можно проводить эксперименты и фиксировать требуемые параметры и в ручном режиме, но в этом случае рекомендуется замедлить эксперимент, например, уменьшив размер добавляемой к массиву последовательности с 10 до 5 элементов.

Требования к проведению экспериментов и содержанию отчета:

Зафиксируйте в отчете данные о текущей конфигурации операционной системы в аспекте управления памятью: 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.

Параметры по оперативной памяти получим с помощью команды «cat /proc/meminfo»

```
MemTotal:        4024160 kB
MemFree:         2698972 kB
MemAvailable:    3071072 kB
Buffers:           51452 kB
Cached:           500472 kB
SwapCached:            0 kB
Active:           782968 kB
Inactive:         330132 kB
Active(anon):     562252 kB
Inactive(anon):      620 kB
Active(file):     220716 kB
Inactive(file):   329512 kB
Unevictable:           0 kB
Mlocked:               0 kB
SwapTotal:       2097148 kB
SwapFree:        2097148 kB
Dirty:              4684 kB
Writeback:             0 kB
AnonPages:        561172 kB
Mapped:           193500 kB
Shmem:              1700 kB
KReclaimable:      55856 kB
Slab:             140620 kB
SReclaimable:      55856 kB
SUnreclaim:        84764 kB
KernelStack:        8800 kB
PageTables:        12344 kB
NFS_Unstable:          0 kB
Bounce:                0 kB
WritebackTmp:          0 kB
CommitLimit:     4109228 kB
Committed_AS:    3667768 kB
VmallocTotal:   34359738367
VmallocUsed:       27228 kB
VmallocChunk:          0 kB
Percpu:            22272 kB
HardwareCorrupted:     0 kB
AnonHugePages:         0 kB
ShmemHugePages:        0 kB
ShmemPmdMapped:        0 kB
FileHugePages:         0 kB
FilePmdMapped:         0 kB
CmaTotal:              0 kB
CmaFree:               0 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
Hugetlb:               0 kB
DirectMap4k:      176048 kB
DirectMap2M:     4018176 kB
```

Параметры по файлу подкачки получим выполнив «cat /proc/swaps»

```
Filename                                Type            Size    Used    Priority
/swapfile                               file            2097148 0       -2
```

Размер страницы виртуальной памяти «getconf PAGE_SIZE» : 4096

MemTotal – доступный объем оперативной памяти. Часть физически доступной памяти резервируется во время запуска системы ядром и не входит в указанный здесь объем, поэтому проверим с помощью команды ***grep Memory: /var/log/dmesg*** сколько памяти зарезервировало ядро. Получаем что общий объем оперативной памяти с учетом ядра получается 4193832 килобайт.

```
[    0.089749] kernel: Memory: 3967400K/4193832K available (14339K kernel code,
2397K rwdata, 4952K rodata, 2712K init, 4992K bss, 226432K reserved, 0K cma-rese
rved)
```

Swap {Total, Free} ⟾***SwapTotal*** – это общий объем области подкачки (как в разделе подкачки, так и в swap-файлах, если они используются). Как и в случае с ОЗУ, ядро Linux старается использовать область подкачки максимально эффективно. Так же общий объем раздела/файла подкачки можно узнать с помощью команды ***cat /proc/swaps***. Каждая строка относится к отдельному пространству подкачки, имеющемуся в системе. В данном случае строка всего одна. Поле ***"Type"*** (Тип) говорит, что мы имеем файл (***file***), а не раздел. Поле ***"Filename"*** сообщает, что этот файл находится в директории ***/swapfile*** (на диске, где установлена ОС). Поле ***"Size"*** (размер) показывает размер файла в килобайтах. Поле ***"Used"*** (Использовано) сообщает, сколько килобайт пространства подкачки используется (в данном случае ноль). ***"Priority"*** (Приоритет) сообщает, какое из пространств подкачки Linux использует первым.

Эксперимент №1:
Подготовительный этап:
Создайте скрипт ***mem.bash***, реализующий следующий сценарий. Скрипт 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, например, (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).

Первый этап:
Задача – оценить изменения параметров, выводимых утилитой ***top*** в процессе работы созданного скрипта.

Ход эксперимента:
Запустите созданный скрипт ***mem.bash***. Дождитесь аварийной остановки процесса и вывода в консоль последних сообщений системного журнала. Зафиксируйте в отчете последнюю запись журнала – значения параметров, с которыми произошла аварийная остановка процесса. Также зафиксируйте значение в последней строке файла ***report.log***. Подготовьте две консоли. В первой запустите утилиту ***top***. Во второй запустите скрипт и переключитесь на первую консоль. Убедитесь, что в ***top*** появился запущенный скрипт. Наблюдайте за следующими значениями (и фиксируйте их изменения во времени в отчете):

- значения параметров памяти системы (верхние две строки над основной таблицей);
- значения параметров в строке таблицы, соответствующей работающему скрипту;
- изменения в верхних пяти процессах (как меняется состав и позиции этих процессов).

Проводите наблюдения и фиксируйте их в отчете до аварийной остановки процесса скрипта и его исчезновения из перечня процессов в ***top***.
Посмотрите с помощью команды ***dmesg | grep "mem.bash"*** последние две записи о скрипте в системном журнале и зафиксируйте их в отчете. Также зафиксируйте значение в последней строке файла ***report.log***.

Второй этап:
Задача – оценить изменения параметров, выводимых утилитой ***top*** в процессе работы нескольких экземпляров созданного скрипта.

Ход эксперимента:
Создайте копию скрипта, созданного на предыдущем этапе, в файл ***mem2.bash***. Настройте её на запись в файл ***report2.log***. создайте скрипт, который запустит немедленно друг за другом оба скрипта в фоновом режиме. Подготовьте две консоли. В первой запустите утилиту ***top***. Во второй запустите созданный перед этим скрипт и переключитесь на первую консоль. Убедитесь, что в top появились ***mem.bash*** и ***mem2.bash***. Наблюдайте за следующими значениями (и фиксируйте их изменения во времени в отчете):

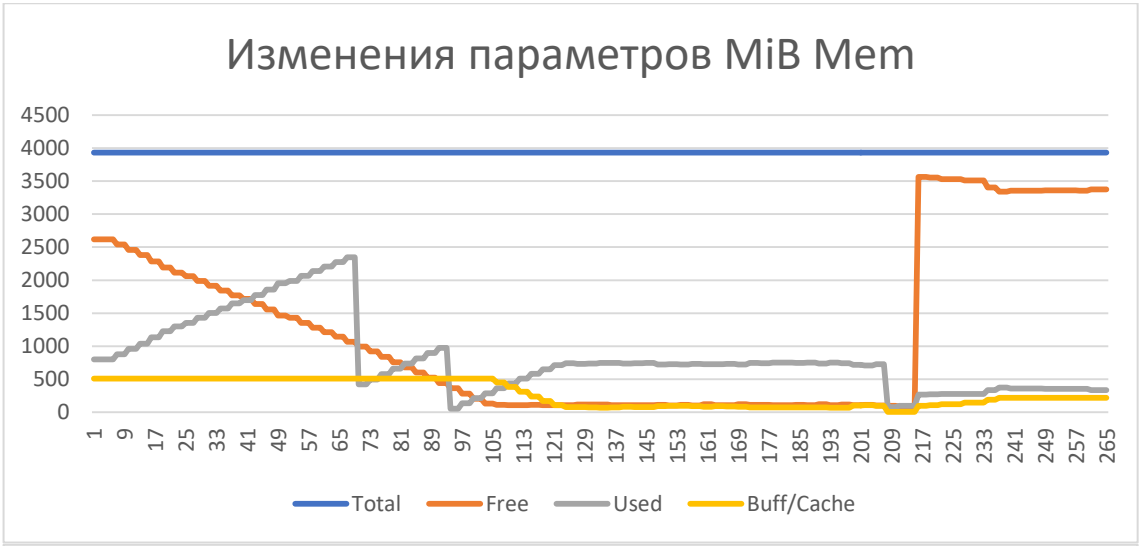
- значения параметров памяти системы (верхние две строки над основной таблицей);
- значения параметров в строке таблицы, соответствующей работающему скрипту;
- изменения в верхних пяти процессах (как меняется состав и позиции этих процессов).

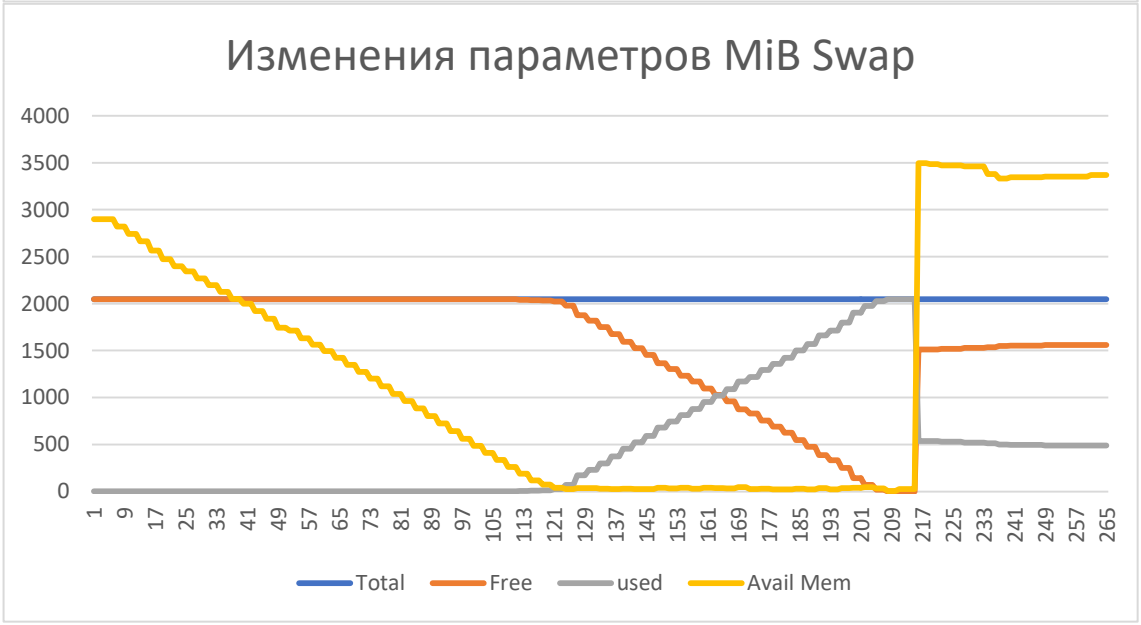
Проводите наблюдения и фиксируйте их в отчете до аварийной остановки последнего из двух скриптов и их исчезновения из перечня процессов в ***top***.

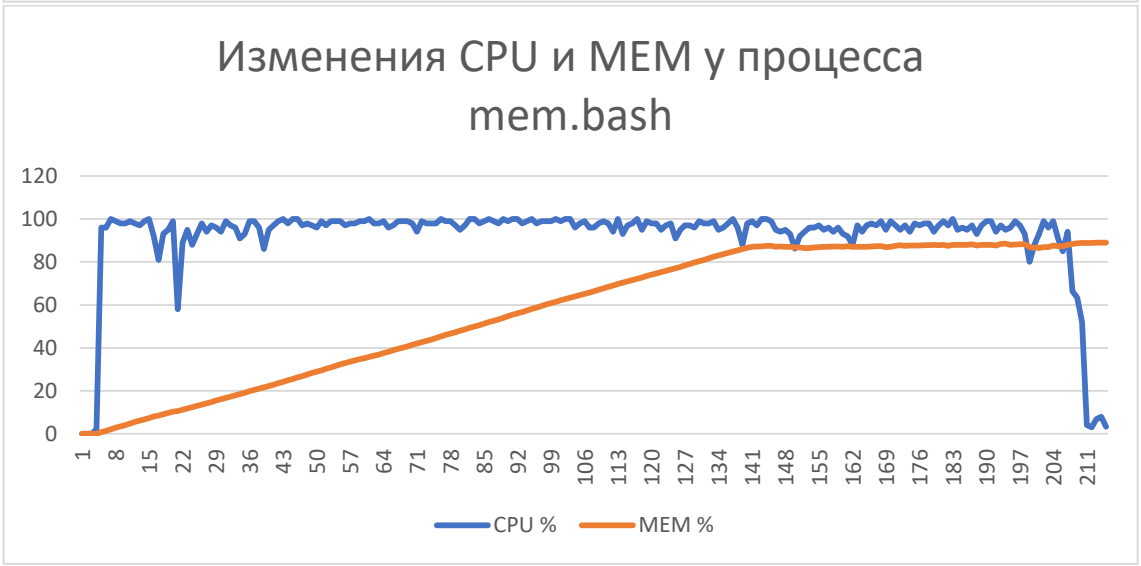
Посмотрите с помощью команды ***dmesg | grep "mem[2]*.bash"*** последние записи о скриптах в системном журнале и зафиксируйте их в отчете. Также зафиксируйте значения в последних строках файлов ***report.log*** и ***report2.log***.

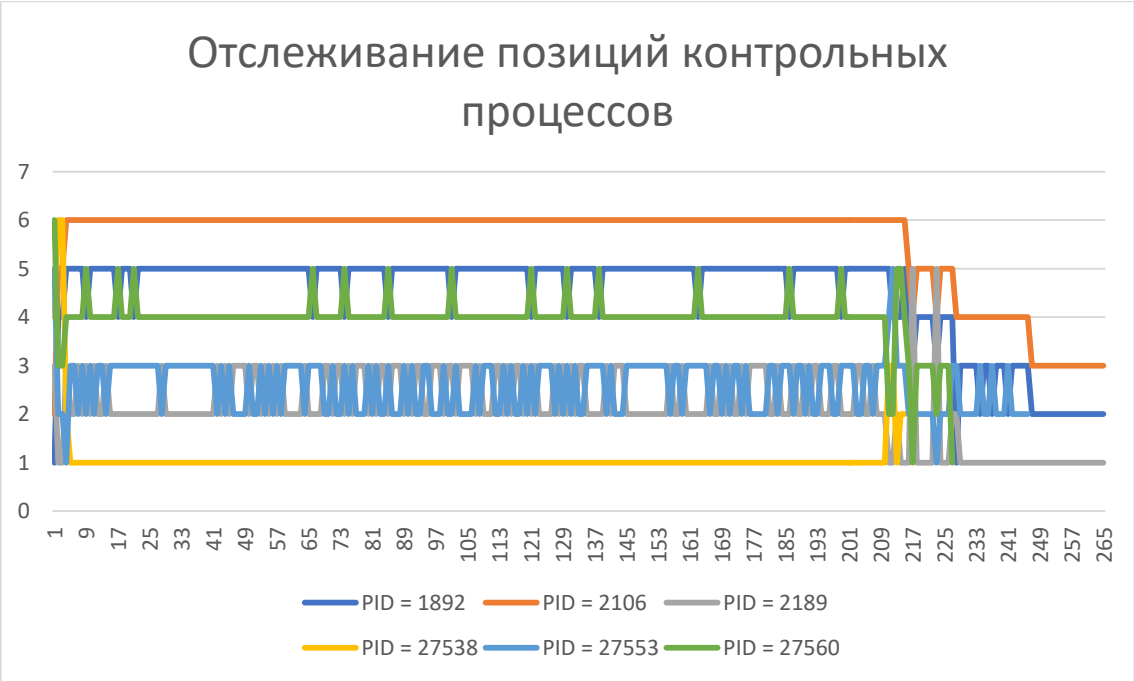
Обработка результатов:
Постройте графики изменения каждой из величин, за которыми производилось наблюдение на каждом из этапов. Объясните динамику изменения этих величин исходя из теоретических основ управления памятью в рамках страничной организации памяти с разделом подкачки. Объясните значения пороговых величин: размер массива, при котором произошла аварийная остановка процесса, параметры, зафиксированные в момент аварийной остановки системным журналом. Сформулируйте письменные выводы.

Выполнение эксперимента №1:
Первый этап:
Все наблюдения и данные были записаны сначала в файл ***HelperReport.log*** с помощью скрипта ***Helper.sh***. Позже все данные об эксперименте были занесены в таблицу ***The monitored parameters.xlsx*** .
С помощью встроенных средств и методов ***Excel*** были построены графики:


Изменения параметров MiB Mem
4500
4000
3500
3000
2500
2000
1500
1000
500
0
1 9 17 25 33 41 49 57 65 73 81 89 97 105 113 121 129 137 145 153 161 169 177 185 193 201 209 217 225 233 241 249 257 265
Total
Free
Used
Buff/Cache


Изменения параметров MiB Swap
4000
3500
3000
2500
2000
1500
1000
500
0
1 9 17 25 33 41 49 57 65 73 81 89 97 105 113 121 129 137 145 153 161 169 177 185 193 201 209 217 225 233 241 249 257 265
Total
Free
used
Avail Mem


Изменения CPU и MEM у процесса mem.bash
120
100
80
60
40
20
0
1 8 15 22 29 36 43 50 57 64 71 78 85 92 99 106 113 120 127 134 141 148 155 162 169 176 183 190 197 204 211
CPU %
MEM %



Также в файле ***HelperReport.log*** были зафиксированы две последние записи системного журнала о скрипте ***mem.bash*** и последнее значение в файле ***report.log***.

```
[ 7486.698864] [  27538]  1000 27538  1286845   895381 10342400   387037             0 mem.bash
[ 7486.698873] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/user.slice/
user-1000.slice/user@1000.service,task=mem.bash,pid=27538,uid=1000
[ 7486.698883] Out of memory: Killed process 27538 (mem.bash) total-vm:5147380kB, anon-rss:3581524kB, file-rss:0kB, shmem-
rss:0kB, UID:1000 pgtables:10100kB oom_score_adj:0
[ 7486.855408] oom_reaper: reaped process 27538 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB


LastIndex: 65600000
```

Обработка результатов:

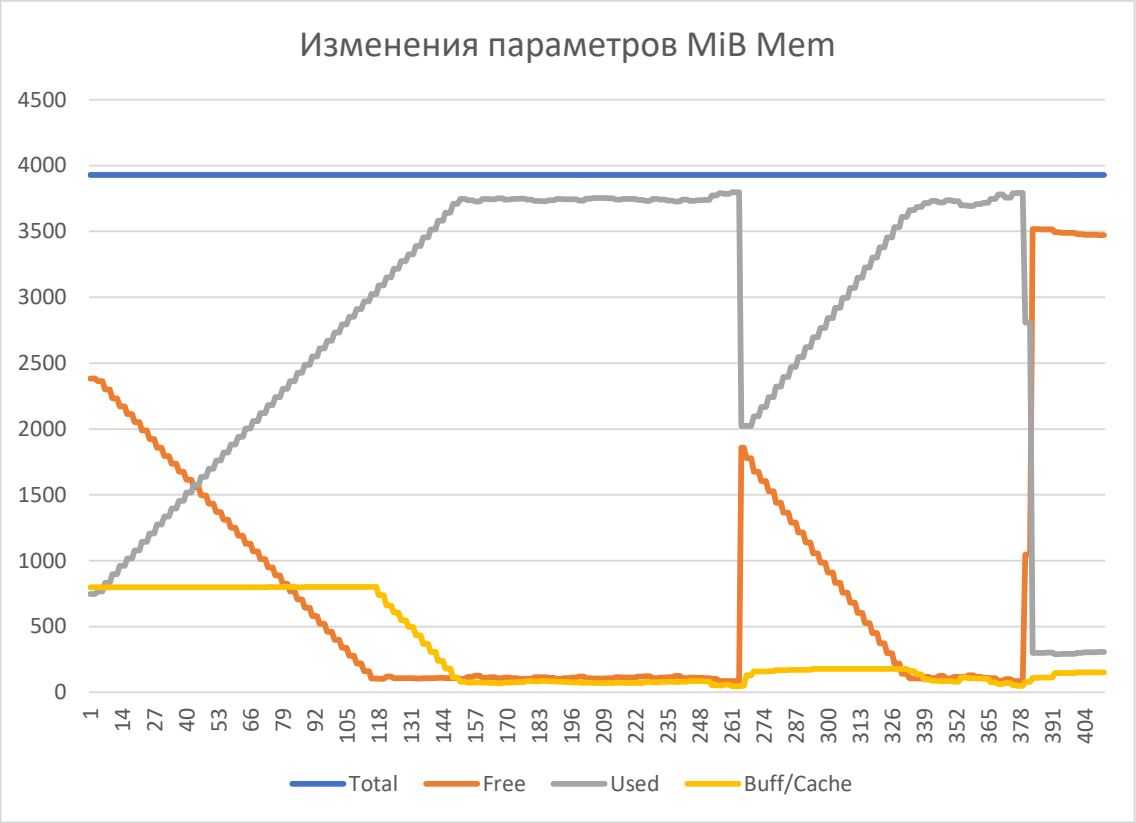
Графики были построены для каждого отслеживаемого параметра.

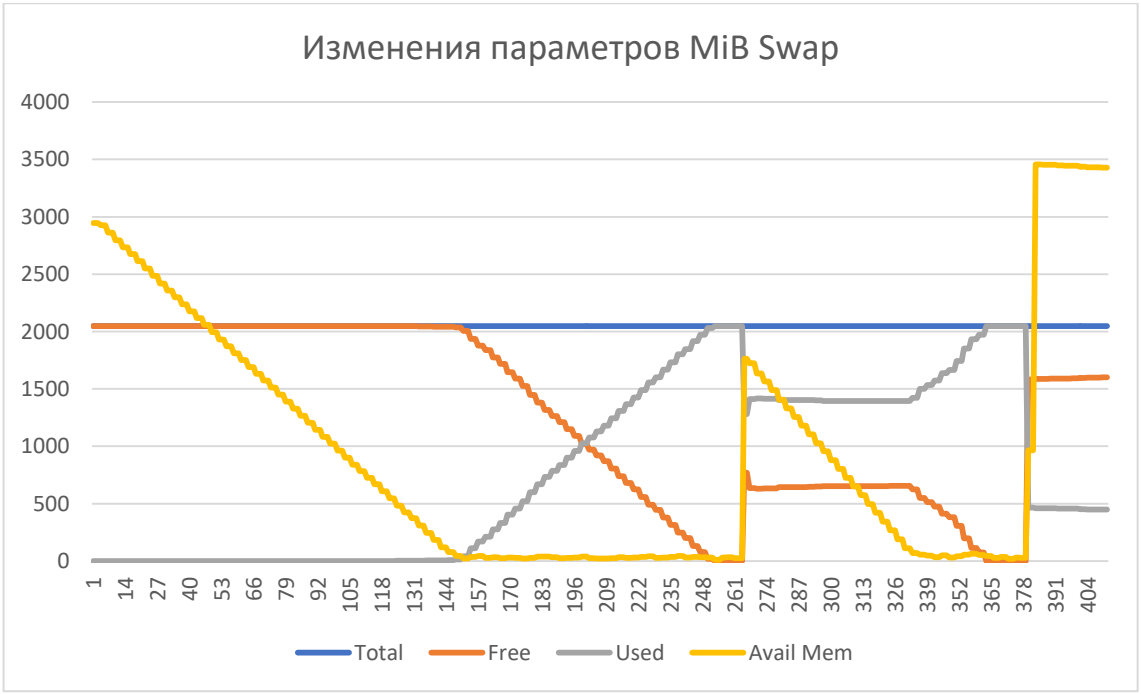
Мы видим на графике *«Изменения параметров MiB Mem»* что кол-во свободной физической оперативной памяти падает. Это происходит из-за того, что процесс ***mem.bash*** требует много ресурсов таких как память и ресурсы процессора, а ОС не может ему отказать из-за относительно высокого приоритета процесса.
На графике *«Изменения параметров MiB Swap»* можно наблюдать резкое уменьшение показателя ***Avail Mem***. Данный параметр характеризует сколько памяти доступно для запуска новых приложений, без подкачки. То есть чем дольше работает ***mem.bash*** тем меньше новых процессов сможет начать выполняться.
Записи системного журнала говорят о том, что процесс ***mem.bash*** был принудительно завершен из-за нехватки памяти.
Размер массива равен *65 600 000* так как именно при таком размере заканчивается вся свободная память.
Из записей системного журнала мы узнаем, что процесс задействовал всю память.
Total-vm = 5 147 380 kB = 4908 MiB ~ *MiB Mem free* + *MiB Swap free*

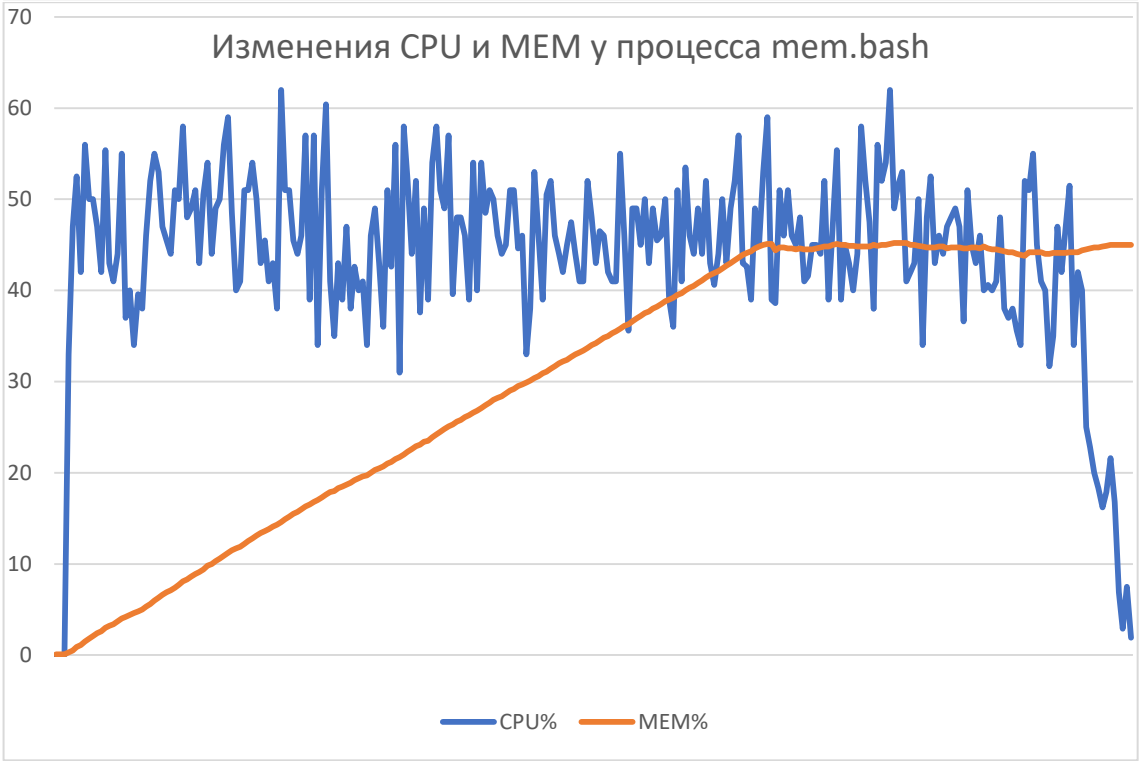
Второй этап:

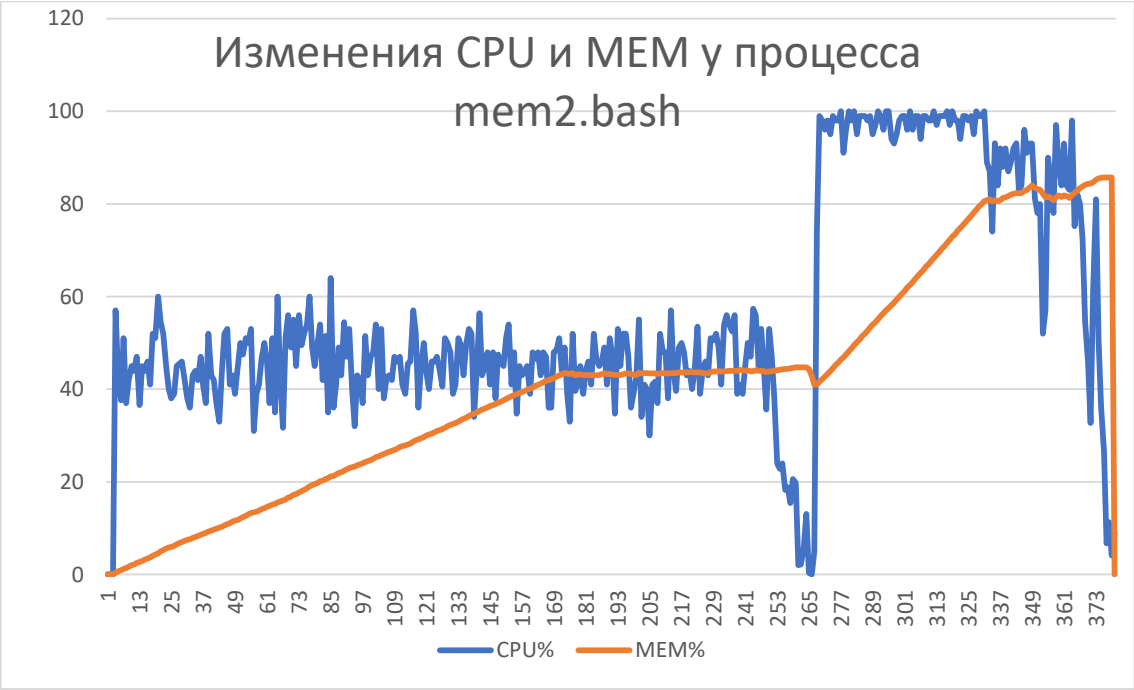
Все наблюдения и данные были записаны сначала в файл ***HelperReport.log*** с помощью скрипта ***Helper.sh***. Позже все данные об эксперименте были занесены в таблицу ***The monitored parameters.xlsx*** для удобства и лучшего понимания.

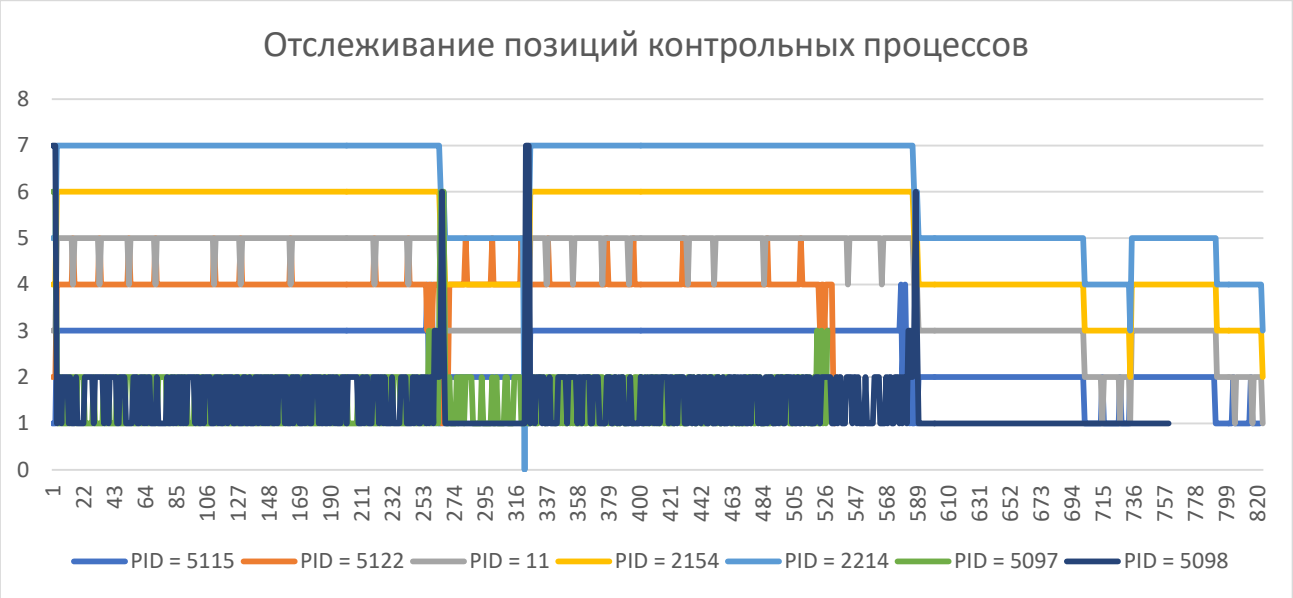
С помощью встроенных средств и методов ***Excel*** были построены графики:




Изменения параметров MiB Swap
4000
3500
3000
2500
2000
1500
1000
500
0
1
14
27
40
53
66
79
92
105
118
131
144
157
170
183
196
209
222
235
248
261
274
287
300
313
326
339
352
365
378
391
404
Total
Free
Used
Avail Mem


Изменения CPU и MEM у процесса mem.bash
70
60
50
40
30
20
10
0
CPU%
MEM%



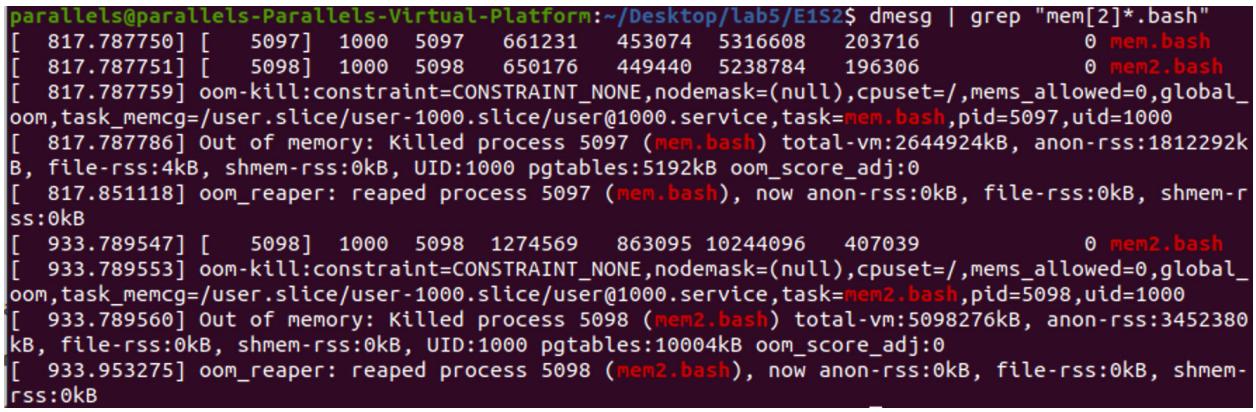


Также в файле ***HelperReport.log*** были зафиксированы последнее значение в файле ***report.log***, последнее значение в файле ***report2.log***, последние записи системного журнала о скриптах ***mem.bash*** и ***mem2.bash***.

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/lab5/E1S2$ dmesg | grep "mem[2]*.bash"
[  817.787750] [   5097]  1000  5097   661231   453074  5316608   203716             0 mem.bash
[  817.787751] [   5098]  1000  5098   650176   449440  5238784   196306             0 mem2.bash
[  817.787759] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/user.slice/
user-1000.slice/user@1000.service,task=mem.bash,pid=5097,uid=1000
[  817.787786] Out of memory: Killed process 5097 (mem.bash) total-vm:2644924kB, anon-rss:1812292kB, file-rss:4kB, shmem-
rss:0kB, UID:1000 pgtables:5192kB oom_score_adj:0
[  817.851118] oom_reaper: reaped process 5097 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB
[  933.789547] [   5098]  1000  5098  1274569   863095 10244096   407039             0 mem2.bash
[  933.789553] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/user.slice/
user-1000.slice/user@1000.service,task=mem2.bash,pid=5098,uid=1000
[  933.789560] Out of memory: Killed process 5098 (mem2.bash) total-vm:5098276kB, anon-rss:3452380kB, file-rss:0kB, shmem-
rss:0kB, UID:1000 pgtables:10004kB oom_score_adj:0
[  933.953275] oom_reaper: reaped process 5098 (mem2.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB



Last Index of Mem1: 33600000
Last Index of Mem2: 65000000
```



Обработка результатов:

Результаты схожи с первым этапом. Можно добавить, что размер массива у процесса ***mem.bash*** равен *33 600 000* так как в этот момент времени заканчивается свободная память. На прошлом этапе мы выяснили, что памяти хватает только на массив размером до *65 600 00,* а *33 600 000* примерно составляет половину от *65 600 00.* После завершения процесса ***mem.bash*** освобождается место в памяти, поэтому процесс ***mem2.bash*** может продолжить свою работу до того, как его массив не станет размером *65 000 000*.

Эксперимент №2

Подготовительный этап:

Создайте копию скрипта ***mem.bash*** в файл ***newmem.bash***. Измените копию таким образом, чтобы она завершала работу, как только размер создаваемого массива превысит значение *N*, передаваемое в качестве параметра скрипту. Уберите запись данных в файл.

Основной этап:

Задача – определить граничные значения потребления памяти, обеспечивающие безаварийную работу для регулярных процессов, запускающихся с заданной интенсивностью.

Ход эксперимента:

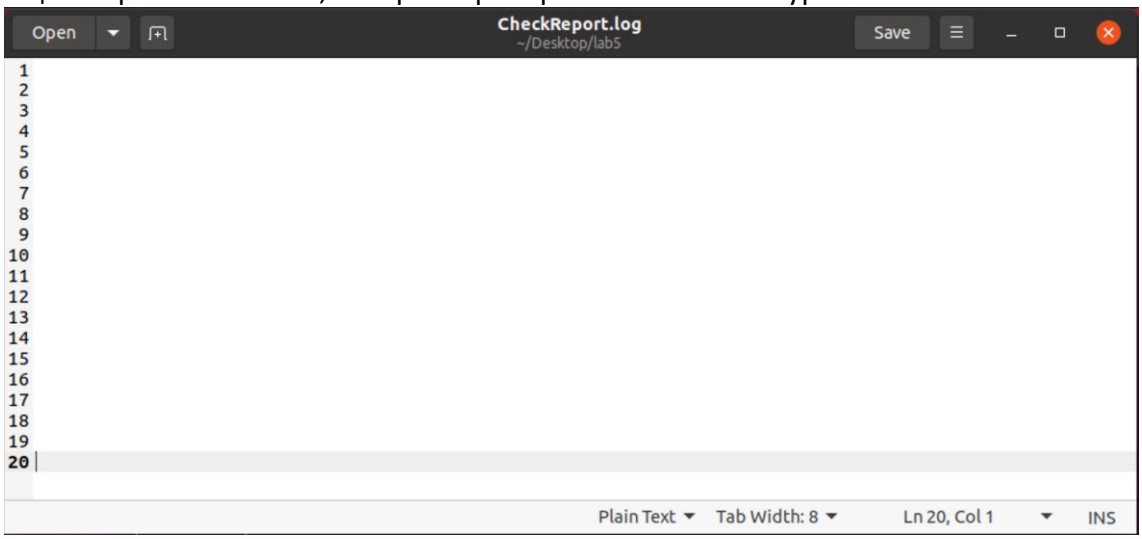
Создайте скрипт, который будет запускать ***newmem.bash*** каждую секунду, используя один и тот же параметр *N* так, что всего будет осуществлено *K* запусков.

Возьмите в качестве значения *N*, величину, в 10 раз меньшую, чем размер массива, при котором происходила аварийная остановка процесса в первом этапе предыдущего эксперимента. Возьмите в качестве *K* значение 10. Убедитесь, что все *K* запусков успешно завершились, и в системном журнале нет записей об аварийной остановке ***newmem.bash***.

Измените значение *K* на 30 и снова запустите скрипт. Объясните, почему ряд процессов завершился аварийно. Подберите такое максимальное значение *N*, чтобы при *K*=30 не происходило аварийных завершений процессов. Укажите в отчете сформулированные выводы по этому эксперименту и найденное значение *N*.

Выполнение эксперимента №2:

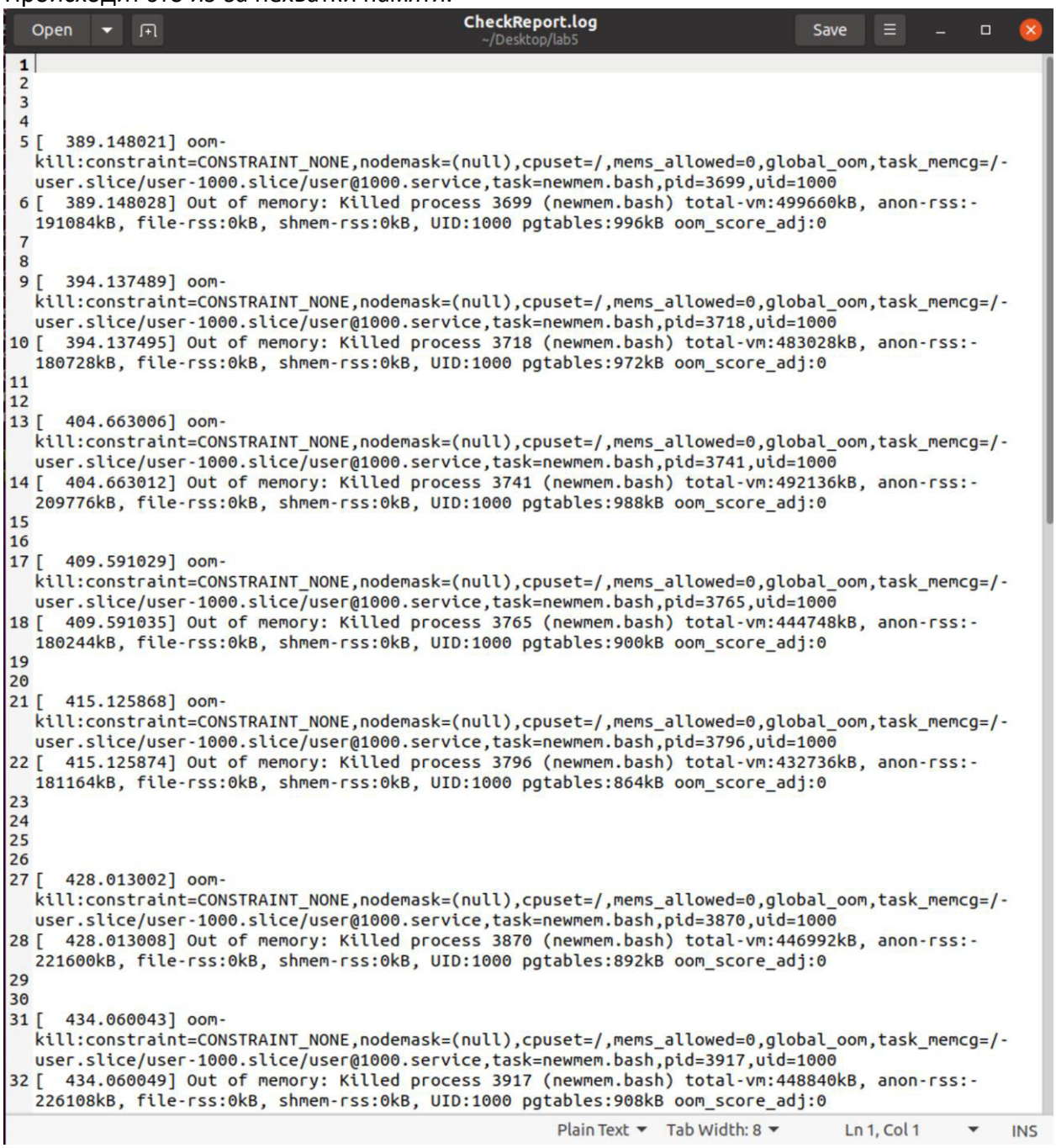
При *N* = 10 и *K* = 6600000 все процессы были завершены успешно. В этом я убеждаюсь с помощью скрипта ***Check.sh***, который проверяет системный журнал на наличие ошибок.



Как видно, скрипт ***Check.sh*** не зафиксировал ошибки в файл ***CheckReport.log***.

Так же я следил за показателями свободной памяти через утилиту ***top*** и делаю вывод, что моментов, когда свободной памяти было мало не было. Было замечено что некоторые ***newmem.bash*** успевали завершить свою работу до того, как запуститься следующий.

При *N* = 30 и *K* = 6600000 не все скрипты успешно завершились. 12 из 30 закончили свою работу аварийно. Какие именно можно узнать из файла ***CheckReport.log***
Происходит это из-за нехватки памяти.



Также было замечено, что при запуске скрипта ***StartMem.sh*** несколько раз через какое-то кол-во попыток процессы перестают падать.
При *N* = 30 и *K* = 2000000 процессы завершаются успешно.