Лабораторная работа №5. Управление памятью в ОС Linux

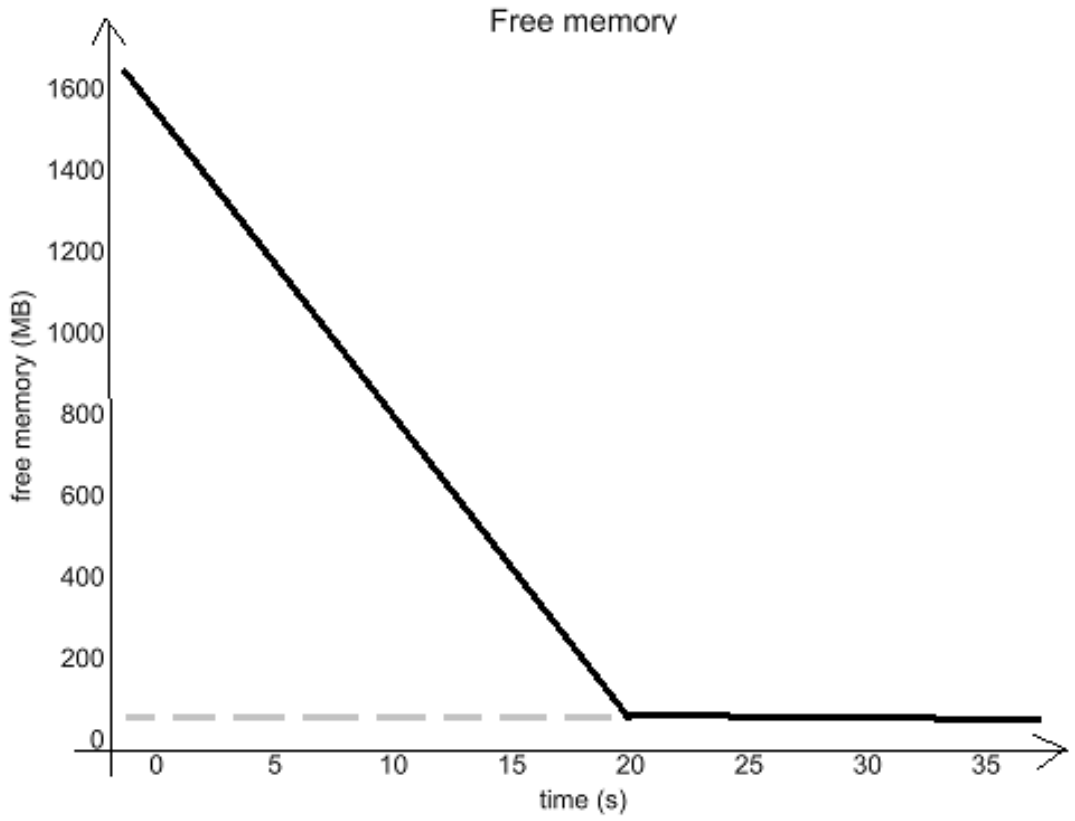
Рассматриваемые вопросы:

1. Использование утилиты top для мониторинга параметров памяти

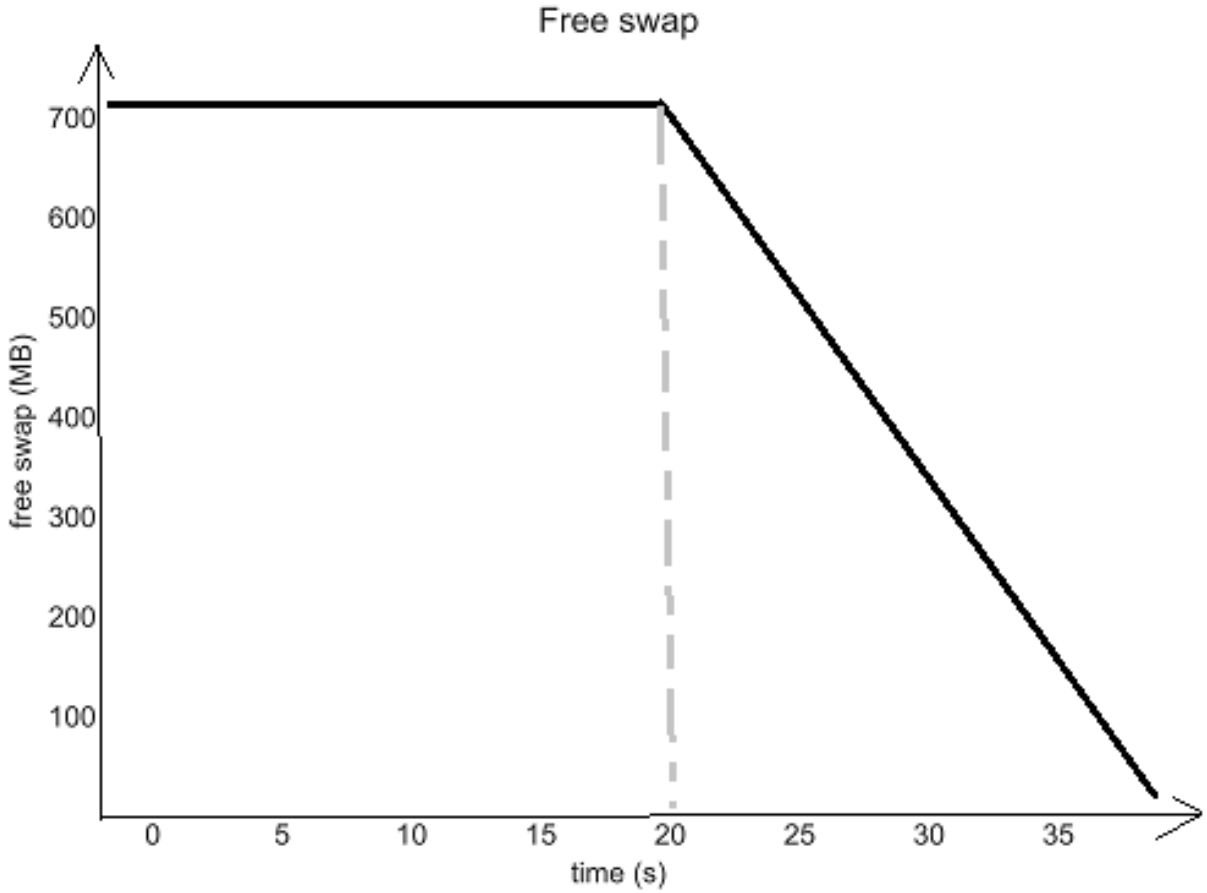
2. Использование имитационных экспериментов для анализа работы механизмов управления памятью.

Эксперимент 1

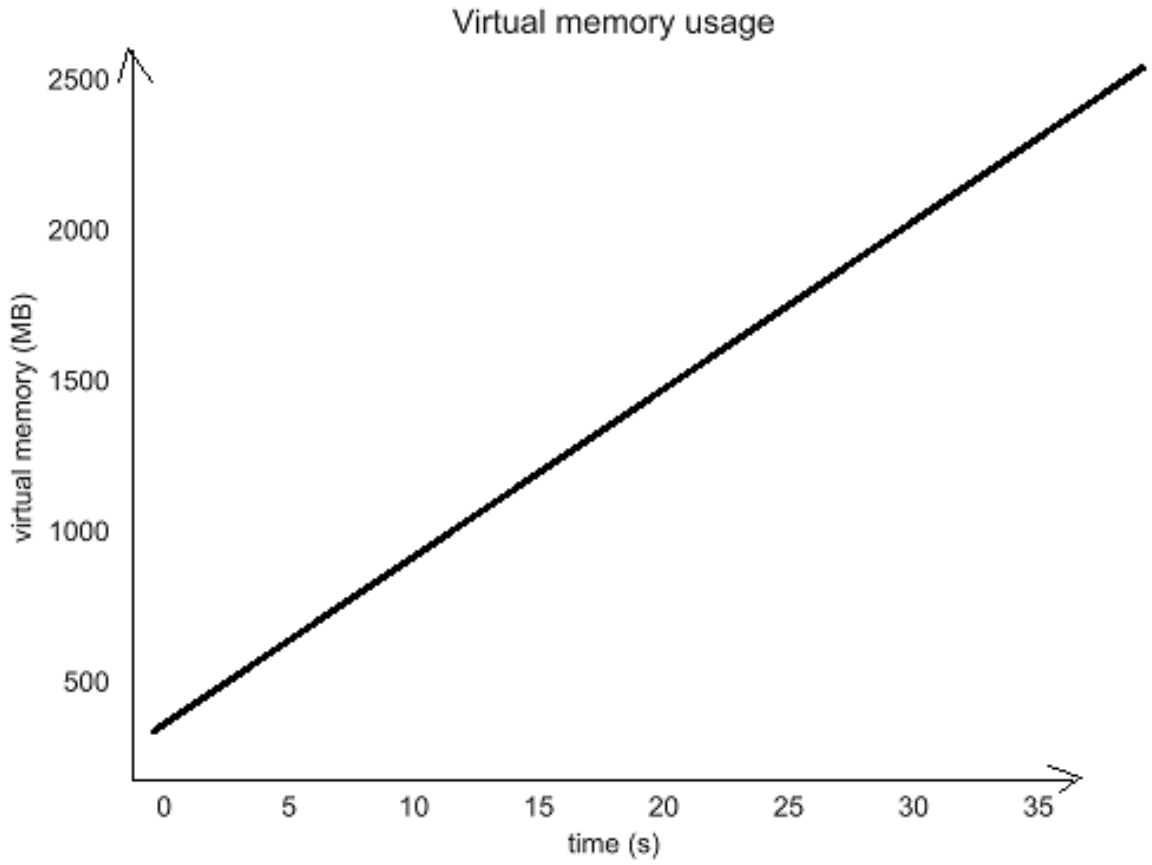
В ходе эксперимента наблюдал за следующими значениями: Free memory, Free swap, Virtual memory, Time. Наблюдения отражены на графиках в зависимости от времени.



Так менялось значение Free memory. Заметим, что примерно на 20 секунде исполнения работы скрипта, то есть когда free memory закончилось, скрипт стал потреблять другую память - free swap, что мы увидим на следующем графике.



Заметим, что использование free swap памяти началось после того, как закончилась free memory. Разберемся, почему так происходит: Free memory - объем свободной физической памяти. Пока она не закончилась, процессу не имеет смысл использовать какую-либо другую память. А когда она кончилась, начинается страничный обмен, для которого как раз и используется free swap (обменная) память.



Видим график использования скриптом виртуальной памяти. В итоге, максимальная использованная виртуальная память равна сумме free memory и free swap. Так, когда заканчивается весь объем возможной виртуальной памяти, скрипт аварийно завершает работу. Можно заметить, что объем использованной виртуальной памяти линейно увеличивается без каких-либо перегибов.

P.S. Еще можно было привести график использования физической памяти (res memory), но это не имеет особого смысла, так как график использования физической памяти - отраженной симметрично по горизонтали график free memory, потому что физическая память - это и есть использованная free memory.

После работы mem1.sh вывод команды dmesg | grep “mem1.sh”: Out of memory: Killed process 1756 (mem1.sh) total-vm:2649176kB, anon-rss: 1674384kB, file-rss:0kB, shmem-rss:0kB, UID:0

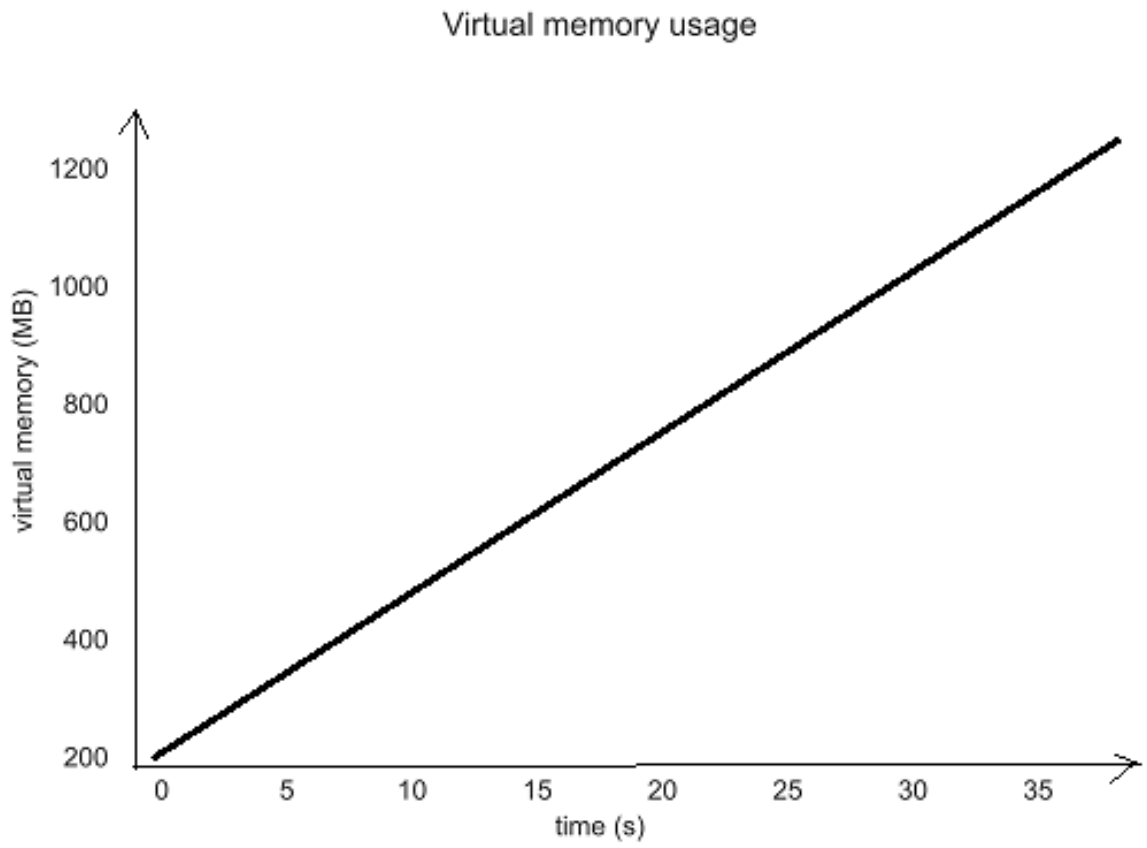
Oom_reaper: reaped process 1756 (mem1.sh), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Последняя строка в report1.log: 3100000 (соответствует примерному количеству элементов в массиве)

Теперь **второй этап**, запуск одновременно mem1.sh и mem2.sh (командой bash memexecutor.sh):

Наблюдал за теми же параметрами, но у двух процессов.

Рассмотрим сначала **mem2.sh**

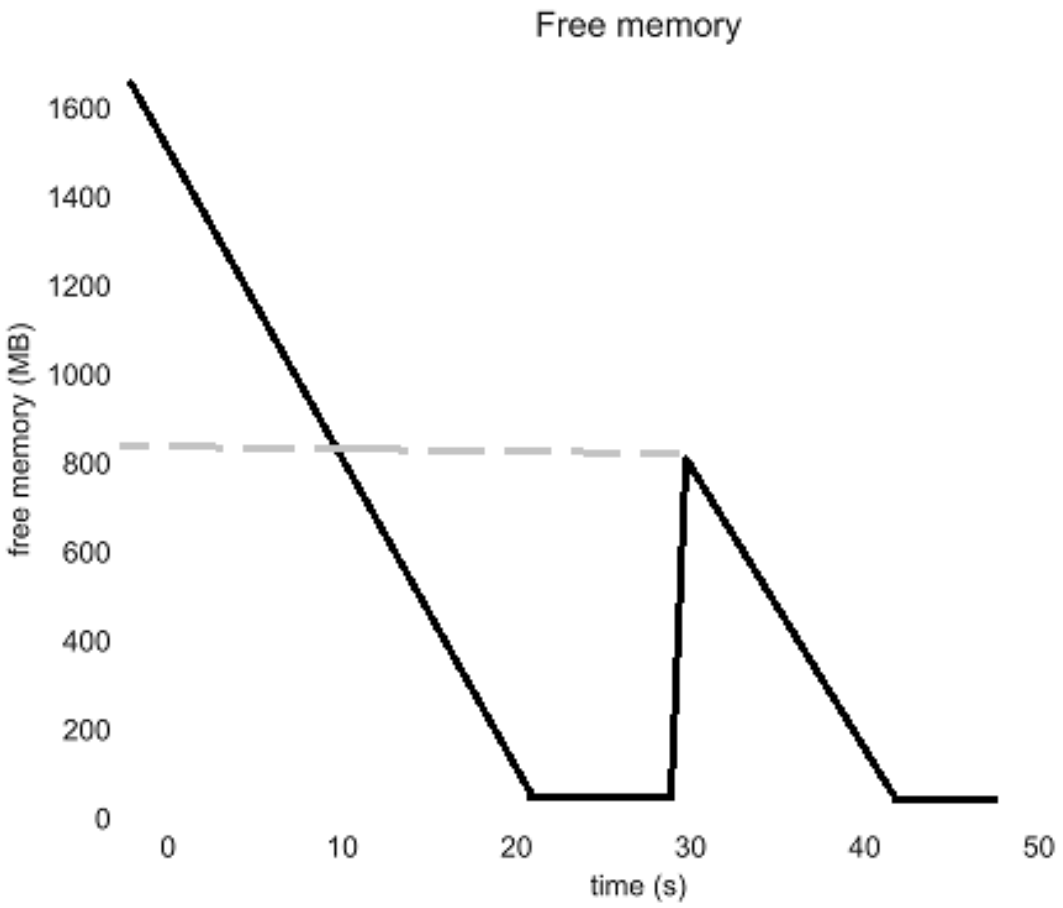


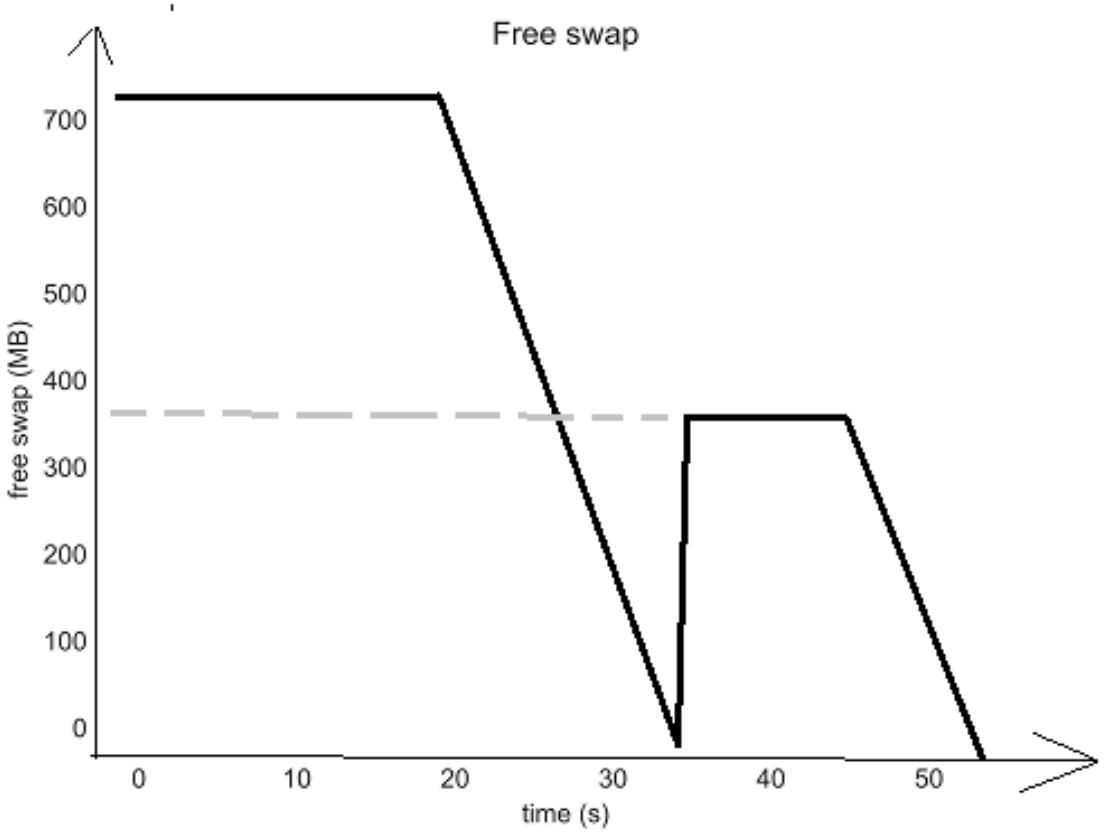
В целом, ситуация аналогична графикам из первого этапа для mem1.sh, поэтому приведу только один график для использования виртуальной памяти. Изменилось только количество использованной памяти (пропорционально, примерно в 2 раза, так как теперь два одинаковых процесса работают одновременно).

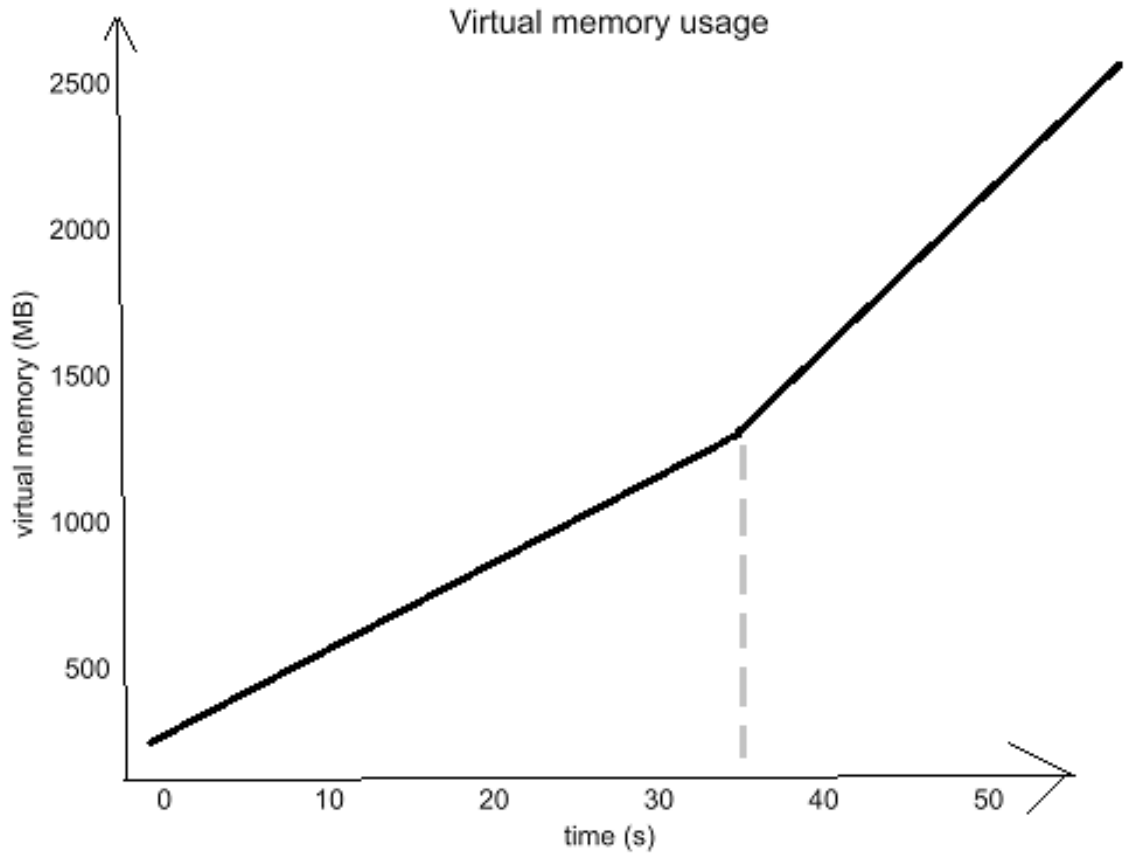
Скрипт завершил работу на 40 секунде исполнения, когда суммарная использованная двумя процессами виртуальная память достигла максимума. Логичное, вполне ожидаемое поведение.

Последняя строка в файле report2.log: 1500000. Также вполне ожидаемо: в два раза меньше элементов в массиве.

Теперь графики для **mem1.sh**:




Free swap
700
600
500
400
300
200
100
0
free swap (MB)
0
10
20
30
40
50
time (s)



Здесь все становится интереснее. Скрипт mem1.sh не завершает работу одновременно с mem2.sh: он продолжает работу, используя ресурсы, освобожденные после аварийного завершения mem2.sh (примерно на 35 секунде). На графике использования виртуальной памяти видно, что с этого момента процесс начинает потреблять память с большей скоростью: логично - теперь он единственный потребитель ресурсов операционной системы. Так, на 35 секунде графиков free memory и free swap мы видим освобождение ресурсов и их дальнейшее переиспользование процессом mem1.sh.
Последняя строка в report1.log: 3100000 - он создал максимальное число элементов в массиве, как и в первом шаге эксперимента.

Вывод команды dmesg | grep ”mem[1-2]*.sh”:
Out of memory: Killed process 1823 (mem2.sh) total-vm:1448636kB, anon-rss:871948kB, file-rss:0kB, shmem-rss:0kB, UID:0
oom_reaper: reaped process 1823 (mem2.sh), now anon-rss: 0kB, file-rss:0kB, shmem-rss:0kB

То есть mem2.sh использовал примерно половину возможных ресурсов и был аварийно завершен

Out of memory: Killed process 1822 (mem1.sh) total-vm:2660264kB, anon-rss:1686564kB, file-rss:132kB, shmem-rss:0kB, UID:0 oom_reaper: reaped process 1822 (mem1.sh), now anon-rss: 0kB, file-rss:0kB, shmem-rss:0kB

А процесс mem1.sh использовал все ресурсы и только после этого был аварийно завершен

Эксперимент 2

При K=30 и N=3*10^6, очевидно, множество процессов завершится аварийно, так как суммарный объем требуемой памяти в три раза превышает объем виртуальной памяти системы, процессы просто не успеют освободить требуемую память, так как запускаются с задержкой всего в 1 секунду, а время их работы будет больше 40 секунд точно.

Я подбирал значение с шагом 100000. Удалось подобрать N=1700000 (при N=1800000 7 процессов завершились аварийно), вывод dmesg | grep “newmem.sh” был пустым, что свидетельствует об отсутствии аварийных завершений процессов. Это связано с тем, что некоторые процессы успевали освободить требуемую память быстрее, чем у остальных процессов заканчивалась свободная память.