*Данные о текущей конфигурации ОС:*

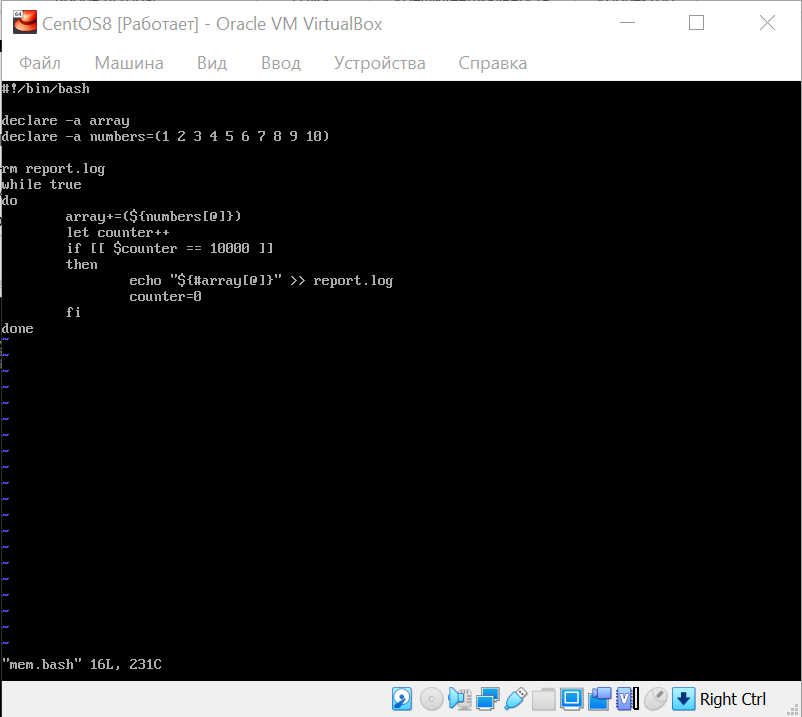
**Отчет**

1. Общий объем оперативной памяти: 1024 Мб
2. Объем раздела подкачки: 820 Мб
3. Размер страницы виртуальной памяти: 4 Кб
4. Объем свободной физической памяти в ненагруженной системе: ~ 572,5 Мб
5. Объем свободного пространства в разделе подкачки в ненагруженной системе: ~ 820 Мб

# Эксперимент I

### Подготовительный этап:

1. Уменьшаем объем оперативной памяти с 4096 Мб на 1024 Мб
2. Скрипт mem.bash, на каждом шаге цикла в конец массива добавляет последовательность из 10 элементов. Далее, каждый 10000-ый шаг добавляет в файл report.log строку с текущим значением размера массива

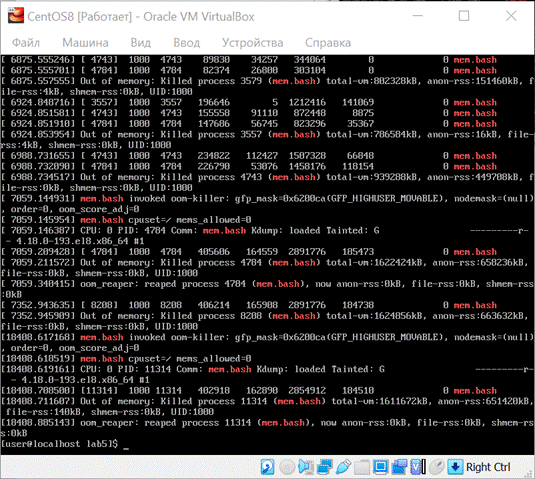


### Первый этап:

1. Значение в последней строке файла report.log: 17800000
2. Данные который получены во время наблюдения (top):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TIME | MEM | VIRT | RES | SHR | CPU | FREE | SWAP |
| 0,54 | 4,4 | 256280 | 36732 | 2756 | 99,9 | 588,6 | 692,7 |
| 2,16 | 16,3 | 356600 | 137052 | 2756 | 99,9 | 493 | 692,7 |
| 3,71 | 28,2 | 455732 | 236052 | 2756 | 99,9 | 394,9 | 692,7 |
| 5,27 | 40,1 | 555656 | 336108 | 2756 | 99,9 | 296,6 | 692,7 |
| 6,83 | 52,2 | 657032 | 437484 | 2756 | 93,8 | 198,9 | 692,7 |
| 8,39 | 64 | 755900 | 536220 | 2756 | 99,9 | 101,8 | 692,7 |
| 11,25 | 76,1 | 935552 | 638316 | 608 | 83,3 | 59,8 | 370,5 |
| 18,24 | 74,6 | 1363892 | 625160 | 580 | 85,7 | 58,4 | 99 |

1. Две записи о скрипте в системном журнале



1. График зависимости времени от памяти полученный на основе данных из (3)

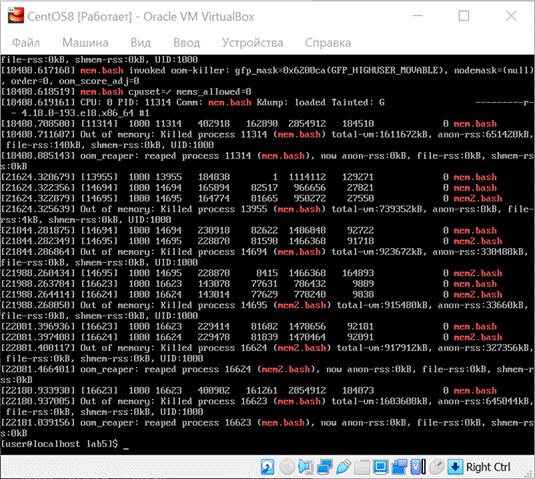
### 

### Второй этап:

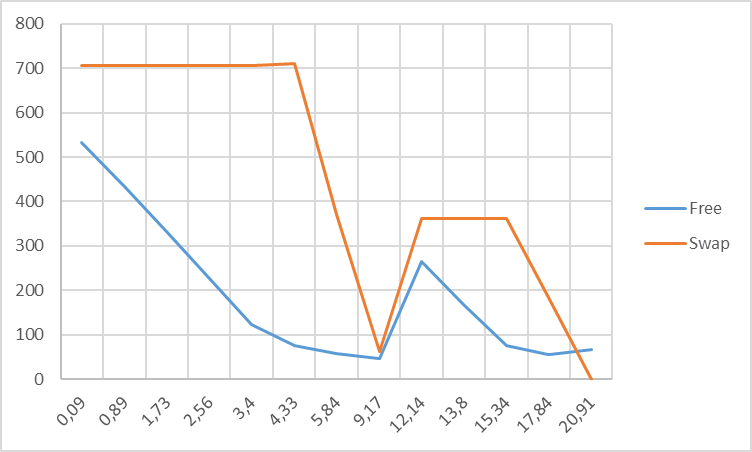
1. Производим аналогичные выше действия только с еще одним скриптом
2. Данные который получены во время наблюдения (top):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| TIME | MEM | VIRT | RES | SHR | CPU | FREE | SWAP |
| 0,09 | 1,1 | 228692 | 9152 | 2904 | 43,8 | 531,8 | 707,1 |
| 0,89 | 7 | 278060 | 58516 | 2904 | 43,8 | 433,8 | 707,1 |
| 1,73 | 13,2 | 330596 | 111052 | 2904 | 46,7 | 333,3 | 707,1 |
| 2,56 | 19,5 | 382868 | 163324 | 2904 | 46,7 | 227,5 | 707,1 |
| 3,4 | 25,8 | 436196 | 216652 | 2904 | 43,8 | 122,9 | 707,1 |
| 4,33 | 32,8 | 494276 | 274732 | 2904 | 52,9 | 76 | 709,9 |
| 5,84 | 37,9 | 588260 | 318088 | 552 | 44,8 | 58 | 370,4 |
| 9,17 | 37,9 | 792596 | 317484 | 592 | 48 | 45,7 | 63 |
| 12,14 | 46,2 | 970136 | 387604 | 1796 | 99,9 | 265,2 | 360,6 |
| 13,8 | 58,7 | 1074812 | 492412 | 1796 | 99,9 | 168,4 | 360,9 |
| 15,34 | 70 | 1169720 | 587188 | 1796 | 99,9 | 76,1 | 360,9 |
| 17,84 | 75,5 | 1327460 | 632792 | 560 | 88,5 | 56,3 | 184,1 |
| 20,91 | 75,1 | 1518596 | 629320 | 556 | 82,6 | 66,6 | 0 |

1. Значение в последней строке файла report.log: 17800000 и 9000000
2. Две записи о скрипте в системном журнале



1. График зависимости времени от памяти полученный на основе данных из (3)



### Наблюдения:

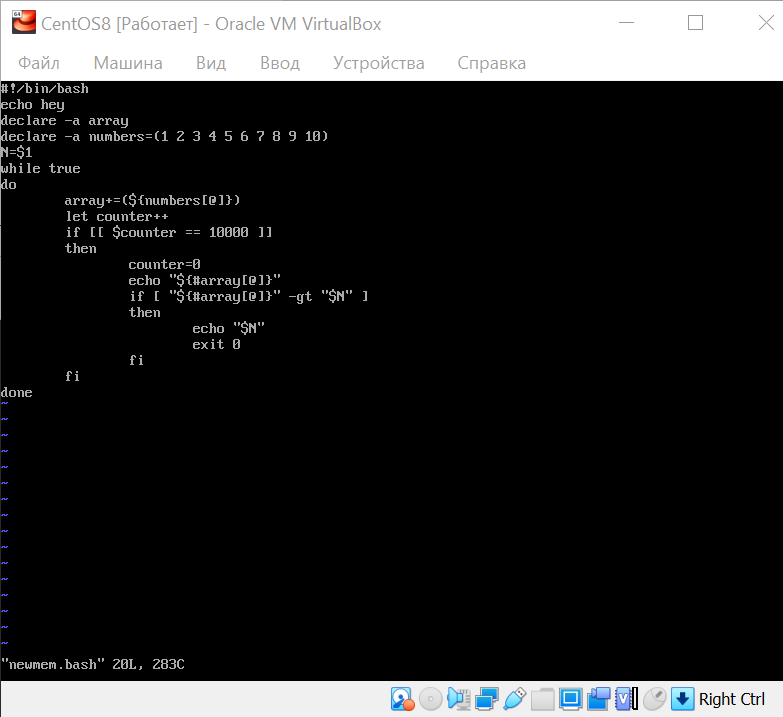
При низкой величине физической памяти (50 – 60 Mб) происходит переход на файлы подкачки. Как только их значение станет критично, то произойдет аварийное завершение программы.

Если запустить 2 скрипта одновременно, то из графика можно заметить, что один из них продолжил работу, это обусловлено тем, что во время

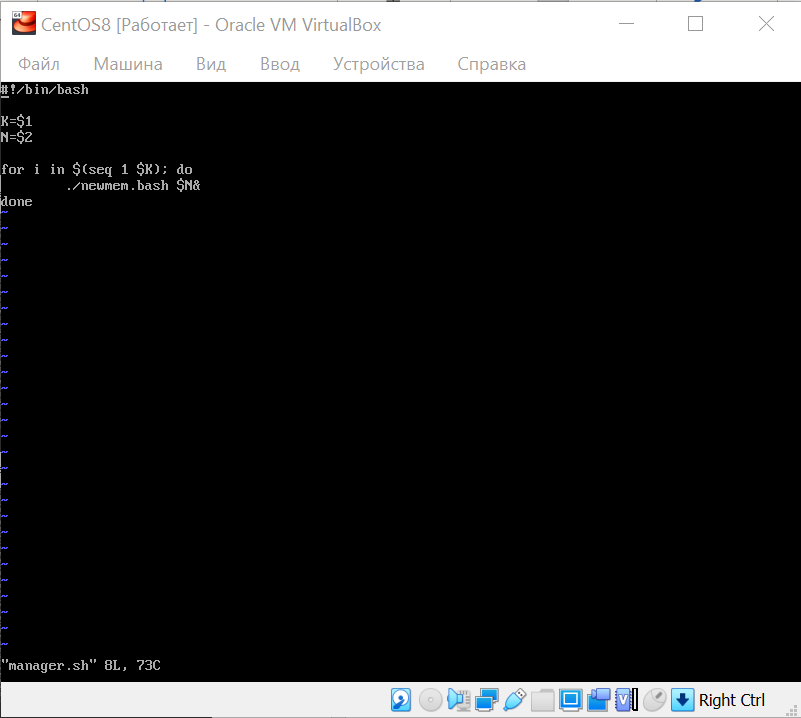
аварийного завершения первого скрипта у нас восстанавливается память, далее идет сценарий как из первого этапа.

**Эксперимент II**

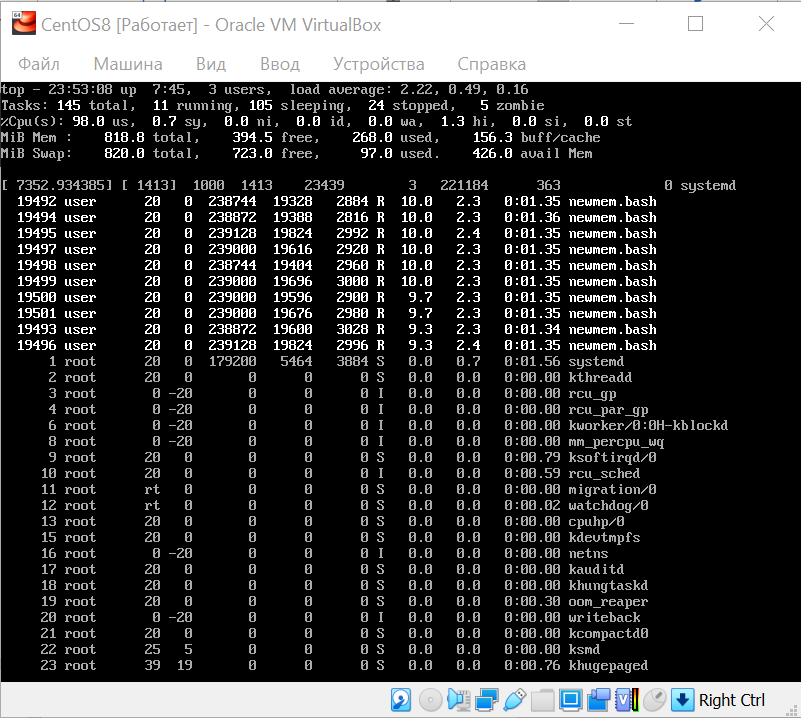
1. Создаем newmem.bash – добавлено условие на прекращение скрипта



1. Был написан менеджер, через который будет производиться запуск

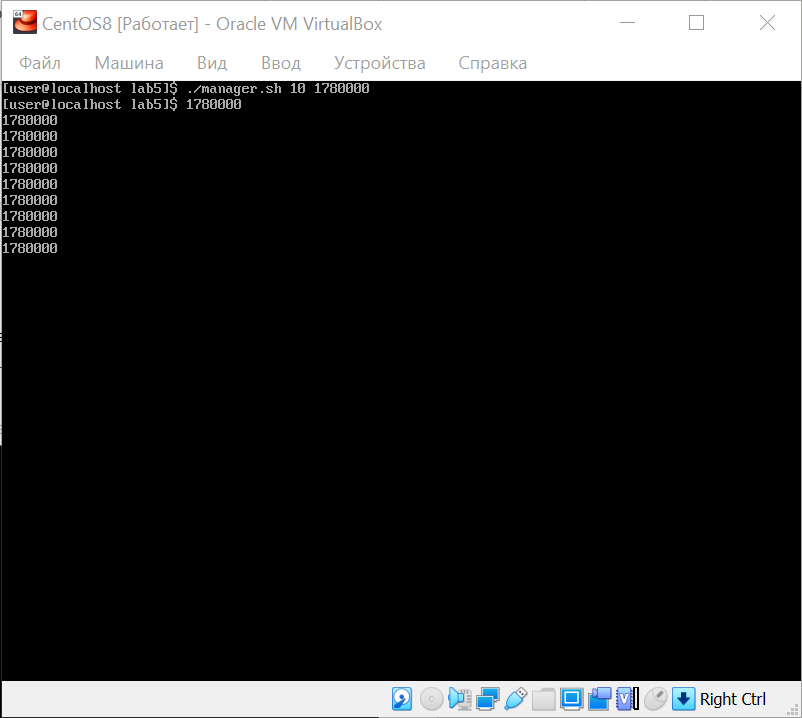


1. Установлено значение К = 10 и N = 1780000
2. Убедимся, что было запущено 10 скриптов

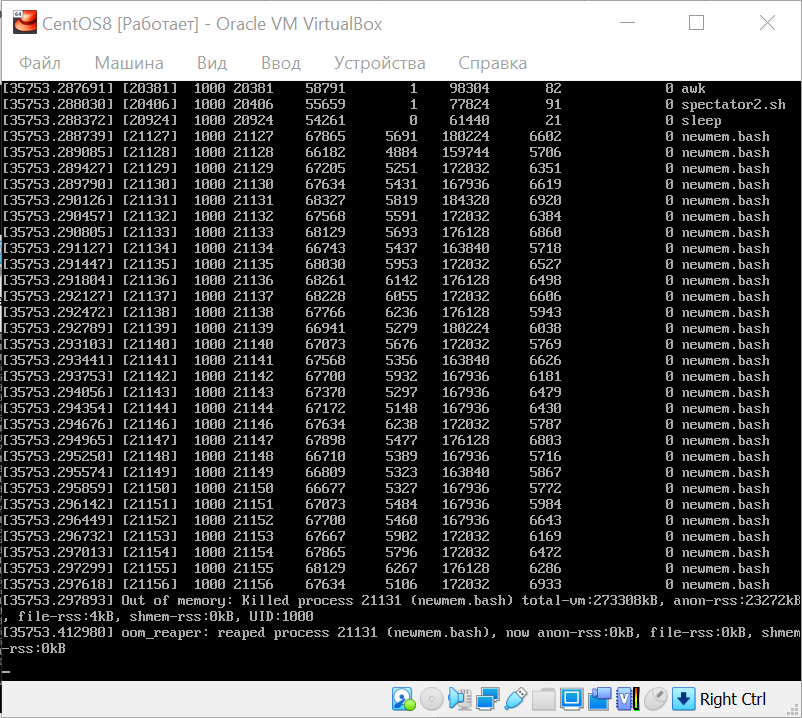


### Наблюдения:

При К = 10 и N = 1780000 программа успешно завершила свое выполнение.

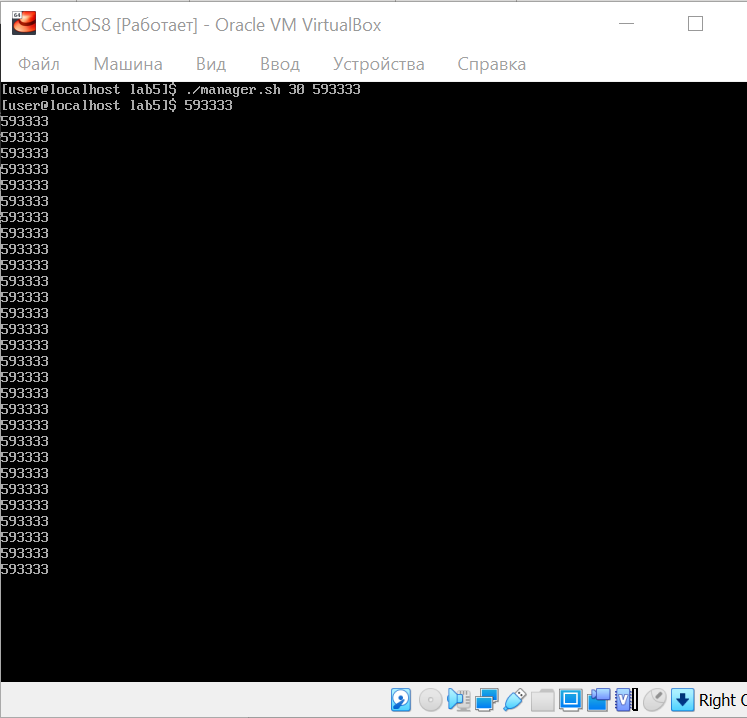


При К = 30 и N = 1780000 программа претерпела аварии, но завершила свое выполнение (при 10 завершенных скриптах)

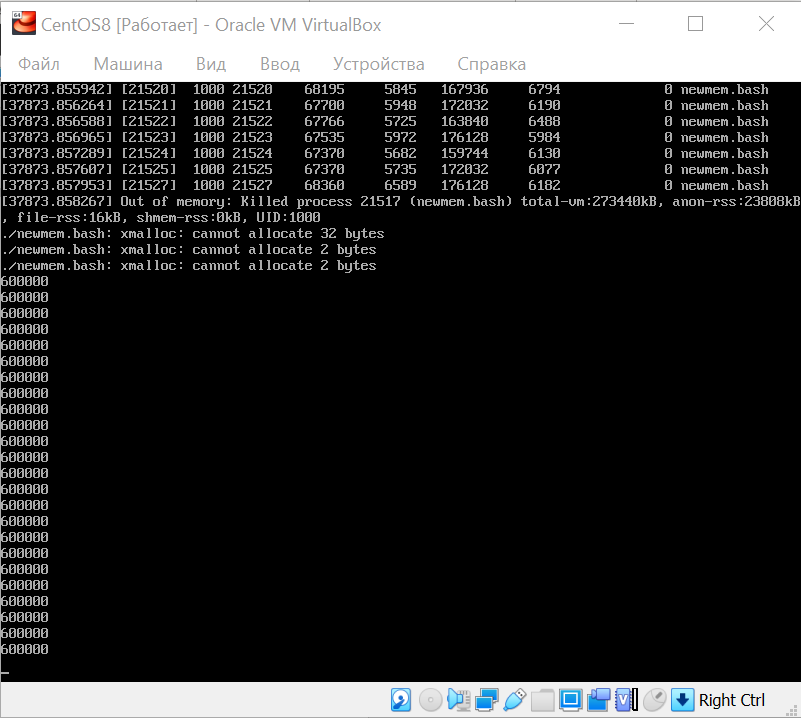


Можно прийти к мысли, что нужно 17800000 разделить на 30 и это будет оптимальное число.

Получаем ~593 333

При К = 30 и N = 593 333 программа успешно завершила свое выполнение. Теория верна

Чтобы окончательно подтвердить теорию, запустим скрипт при N = 600 000, на что получим аварийный конец. Но поскольку отклонение не велико, скрипт был частично выполнен.



**Вывод**

На практике мы убедились, что во время работы с оперативной памятью данные заполняют физическую память, пока не дойдет до критического значения. Далее ОС начинает заполнять файлы подкачки. В случае, когда файлов подкачки будет не хватать – произойдет аварийная остановка процесса. А также количество памяти X, занимаемое одной программой примерно равно N программам, занимающим X/N памяти.