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**

Выполнил студент группы №M3212

***Иванов Анатолий Сергеевич***

Преподаватель:

***Дюкарева Вероника Максимовна***

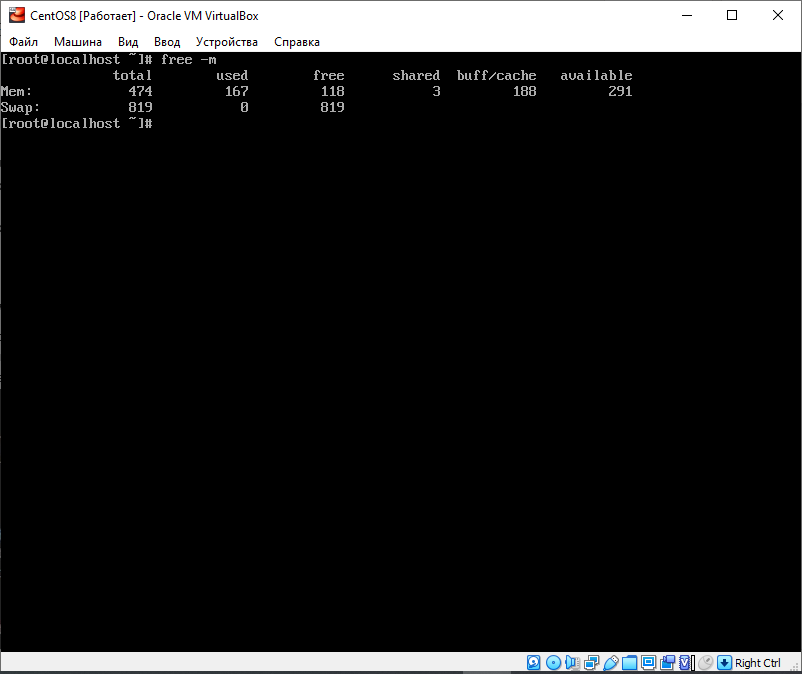
***САНКТ-ПЕТЕРБУРГ***

***2020***

**Отчет**

*Данные о текущей конфигурации ОС:*

1. Общий объем оперативной памяти: 512 Мб
2. Объем раздела подкачки: 819 Мб
3. Размер страницы виртуальной памяти: 4 Кб
4. Объем свободной физической памяти в ненагруженной системе: 118 Мб



1. Объем свободного пространства в разделе подкачки в ненагруженной системе: 819 Мб

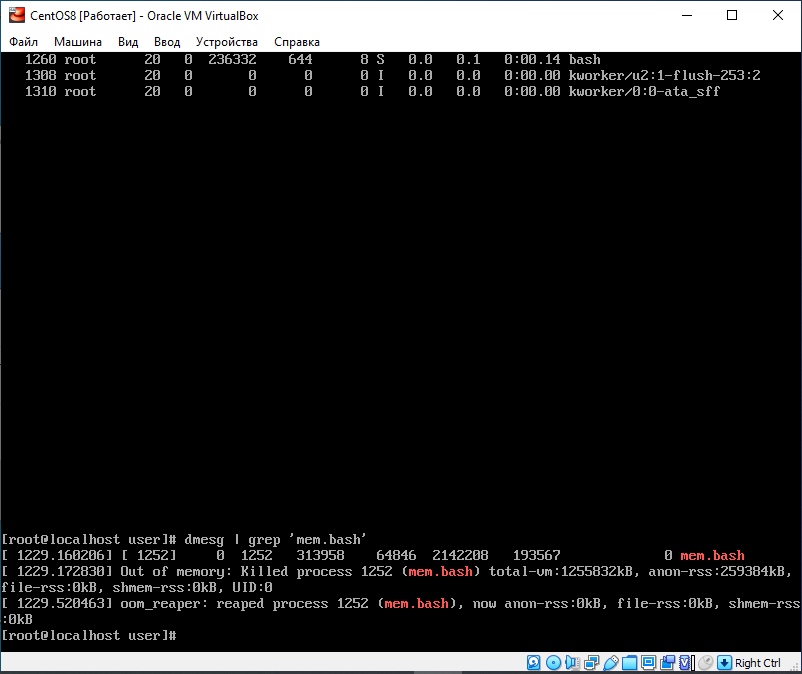
**Эксперимент 1**

*Подготовительный этап:*

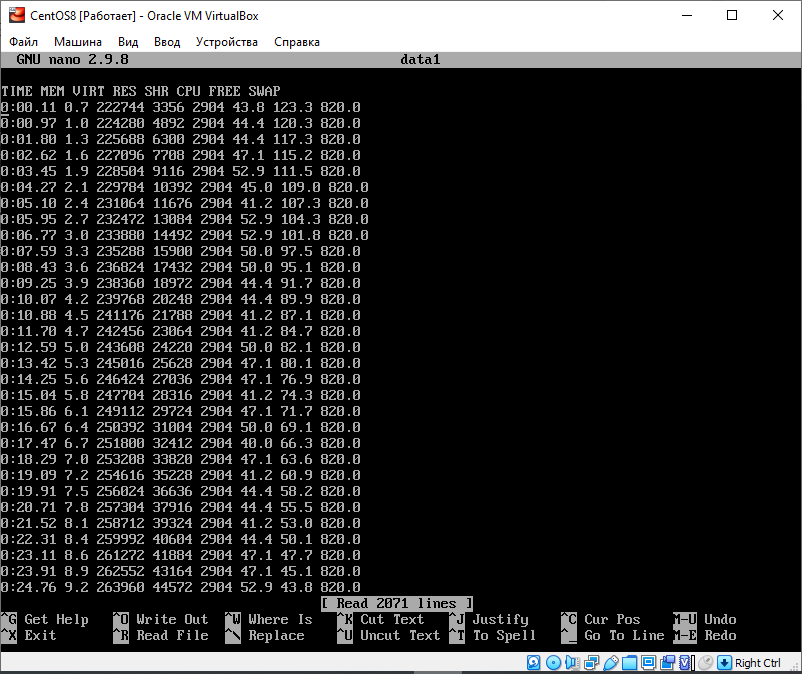
1. Был написан скрипт mem.bash, который на каждом шаге цикла в конец массива добавлял последовательность из 100 элементов. Далее, каждую 10000-ый шаг добавлял в файл report.log строка с текущим значением размера массива

*Первый этап:*

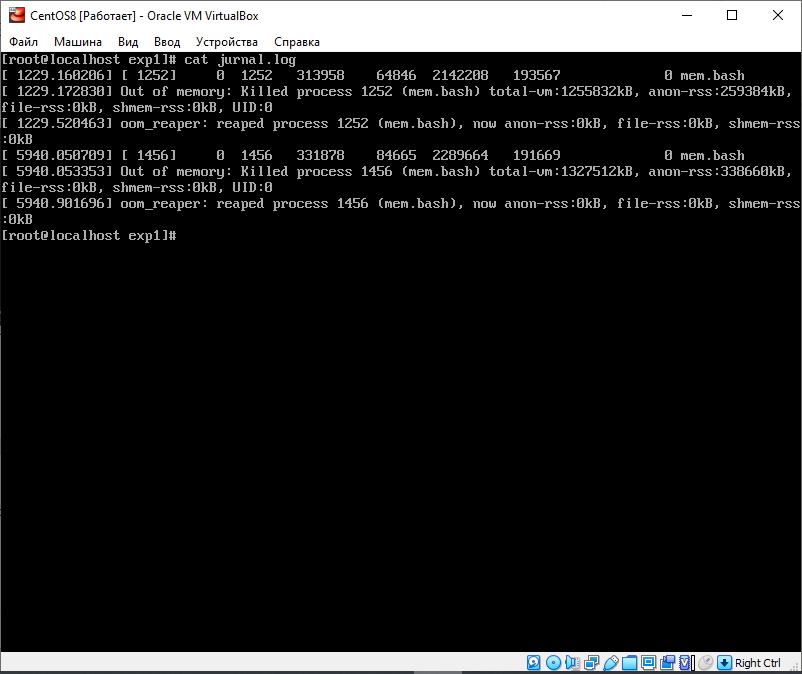
1. Последняя запись журнала – значения параметров, с которыми произошла аварийная остановка процесса.



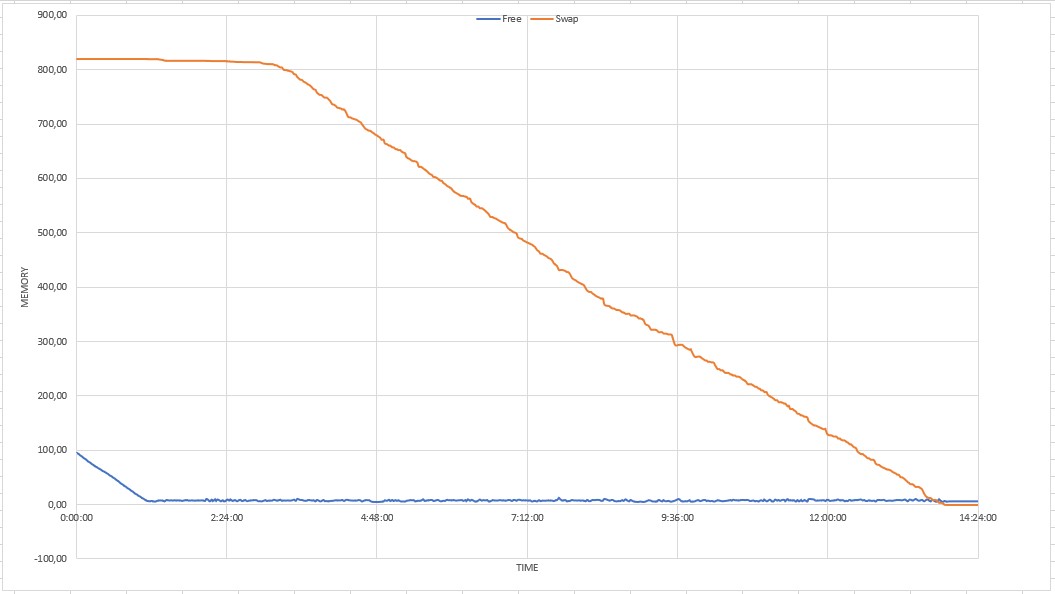
1. Значение в последней строке файла report.log: 30000000
2. Данные которые получены во время наблюдения



1. Две записи о скрипте в системном журнале

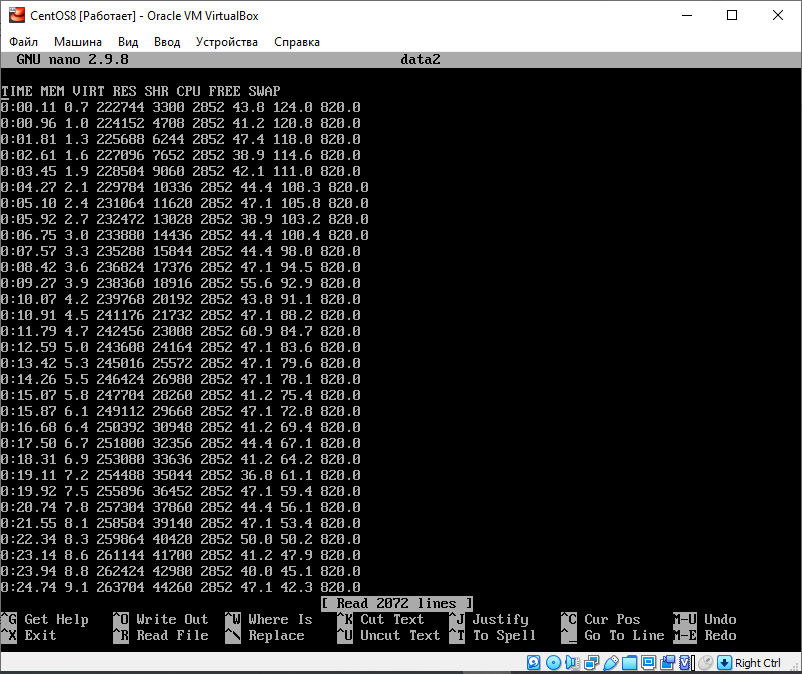


1. График зависимости времени от памяти полученный на основе данных из data1

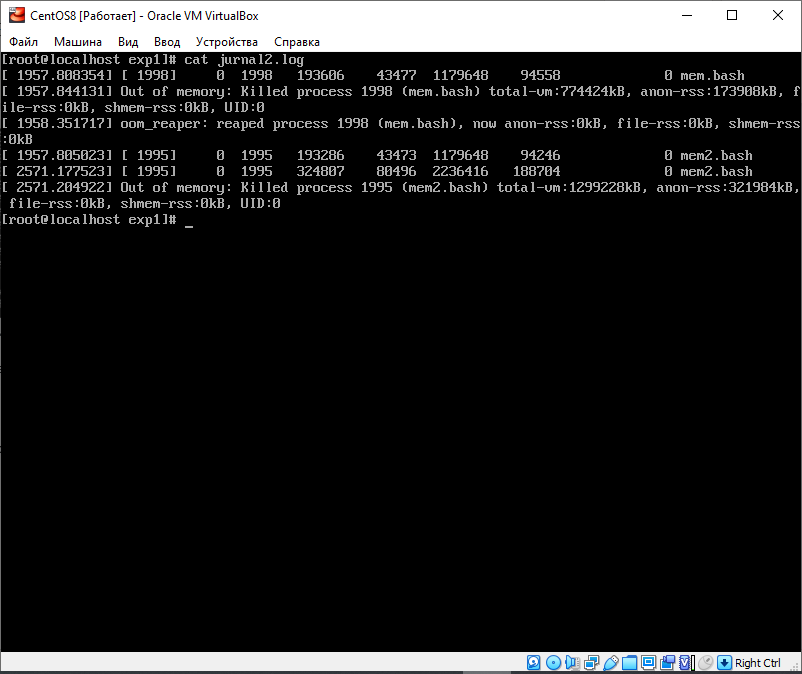


*Второй этап:*

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



1. Значение в последней строке файла report.log и report2.log: 16000000 и 31300000
2. Две записи о скрипте в системном журнале



1. График зависимости времени от памяти полученный на основе данных из data2

0

,

00

100,00

200,00

300,00

400,00

500,00

600,00

700,00

800,00

900,00

0:14:24

1:26:24

2:38:24

3:50:24

5:02:24

6:14:24

7:26:24

8:38:24

9:50:24

MEMORY

TIME

Free

Swap

*Наблюдения:*

При низкой величине физической памяти (6 – 8 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 физическая память, далее идет сценарий как из первого этапа.

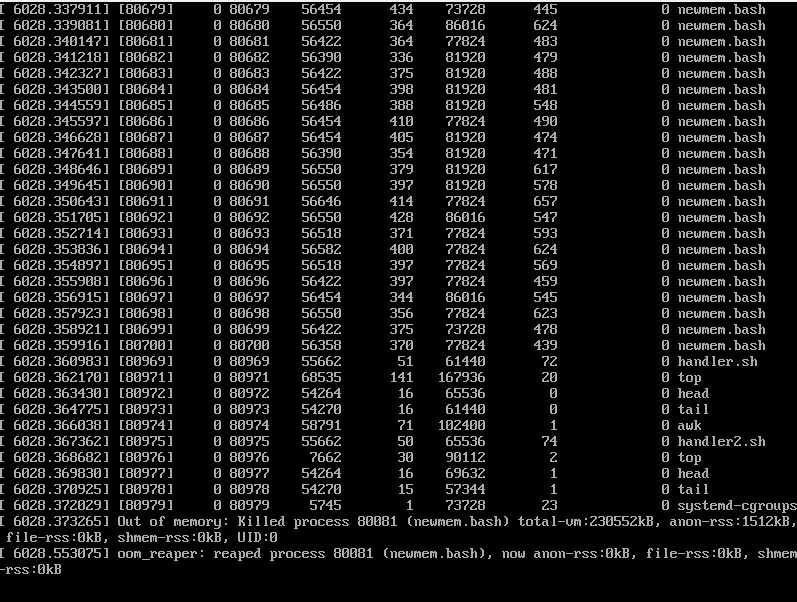
**Эксперимент 2**

1. Подкорректирован newmem.bash – добавлено условие на прекращение скрипта
2. Был написан обработчик, через который будет производиться запуск
3. Установлено значение К = 10 и N = 3000000

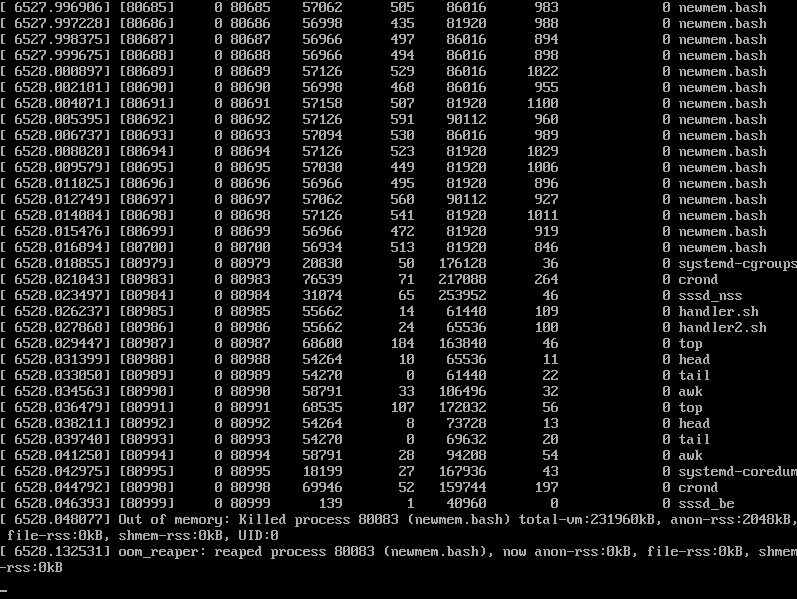
*Наблюдения:*

При К = 10 и N = 3000000 программа успешно завершила свое выполнение.

При К = 30 и N = 3000000 программа аварийно завершила свое выполнение.



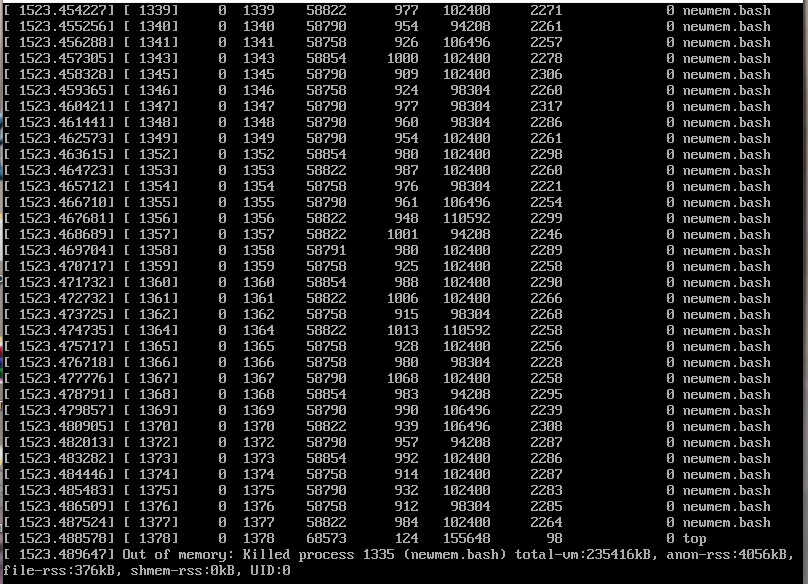
При К = 30 и N = 2900000 программа аварийно завершила свое выполнение.



При К = 30 и N = 1900000 программа успешно завершила свое выполнение.

При К = 30 и N = 2000000 программа успешно завершила свое выполнение**.**

При К = 30 и N = 2100000 программа аварийно завершила свое выполнение.



В итоге можно сказать, что N для успешного выполнения программы находится в диапазоне от 2000000 до 2100000

Все к запусков успешно отработали. Это обуславливается тем, что процессы успевают закончиться раньше,

чем наступает критическое употребление памяти. По мере завершения процессов происходит

высвобождение памяти для следующих процессов. Все выполняется параллельно и успешно

Измените значение K на 30 и снова запустите скрипт. Объясните, почему ряд процессов завершился

аварийно.

При 30 запусках одновременно работают уже больше процессов. Старые не успевают завершиться, как

инициируются новые. Память быстро расходуется. Как результат, многие процессы аварийно завершаются.

Чем освобождают физическую память для других процессов. Также, при запуске новых процессов старые

переходят в раздел подкачки как менее приоритетные и ждут своей очереди на выполнение. Данный

процесс также не бесконечен. Размер буфера подкачки ограничен, и выделяется для помощи оперативной

памяти

**Вывод**

На практике убедился, 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 фалов подкачки будет не хватать произойдет аварийная остановка процесса