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

***Муртазин Рифат Фаритович***

Преподаватель:

***Дюкарева Вероника Максимовна***

***САНКТ-ПЕТЕРБУРГ***

***2020***

**Ход работы**

Данные о текущей конфигурации операционной системы в аспекте управления памятью:

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

**Эксперимент 1**

***Подготовительный этап:***

*Создаю скрипт mem.bash, который имеет массив, бесконечный цикл. На каждой итерации цикла в массив добавляются 10 новых элементов и на каждой 100 000 итерации в файл report.log записывается размер массива.*

***Первый этап:***

Запустил скрипт mem.bash дождался аварийной остановки. Получил

Последняя запись журнала – значения параметров, с которыми произошла аварийная остановка процесса. С помощью команды dmesg | grep "mem.bash"

Изображение выглядит как текст

Автоматически созданное описание

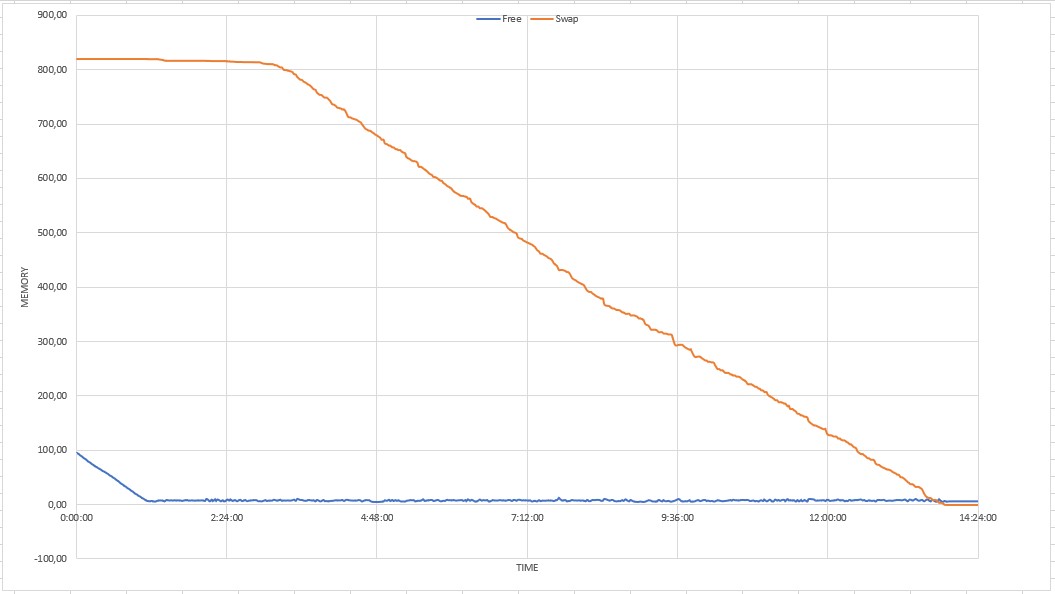
Последняя строка в файле report.log равна 18 600 000

Наблюдения с помощью дополнительного скрипта с выводом в infproc1, который выводит результат наблюдений в infproc1.

Изображение выглядит как текст

Автоматически созданное описание

График зависимости времени от памяти полученный на основе данных из infproc1



***Второй этап:***

Теперь делаю копию скрипта mem.bash и вывожу результаты в report2. Запуская одновременно два скрипта и записываю результат

Производим аналогичные выше действия только с еще одним скриптом

Данные который получены во время наблюдения:

Изображение выглядит как текст

Автоматически созданное описание

Значение в последней строке файла report.log и report2.log: 18 600 000 и 9 200 000

Записи о скрипте в системном журнале:

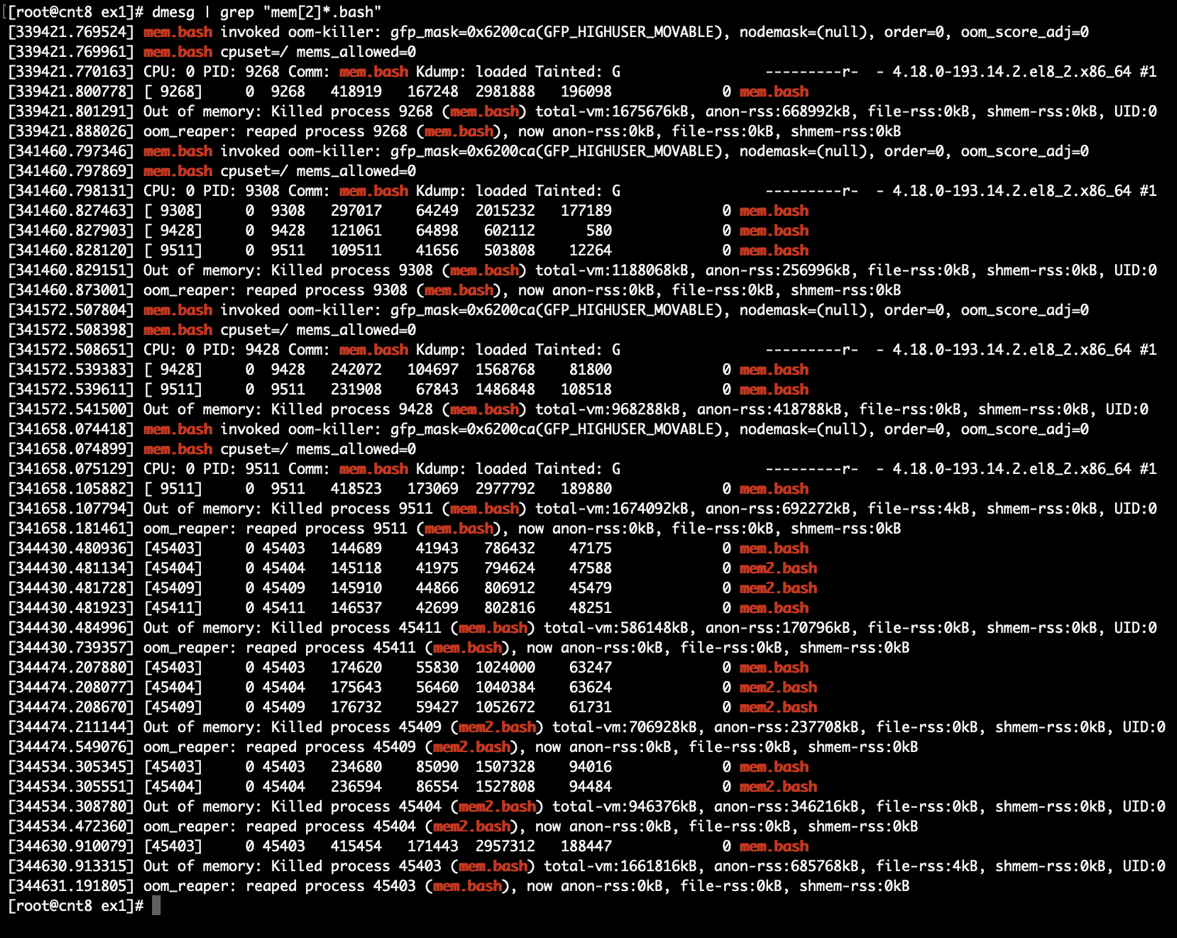


График зависимости времени от памяти полученный на основе данных

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:***

Из-за довольно маленькой величины физической памяти случается переход на файлы подкачки, но как только произойдёт переполнение, произойдет аварийное завершение программы. Также удалось заметить, что, если при запуске 2-ух скриптов одновременно на фоне, то на графике видно, что второй скрипт продолжит работу, потому что во время аварийного завершения первого скрипта появляется свободная физическая память.

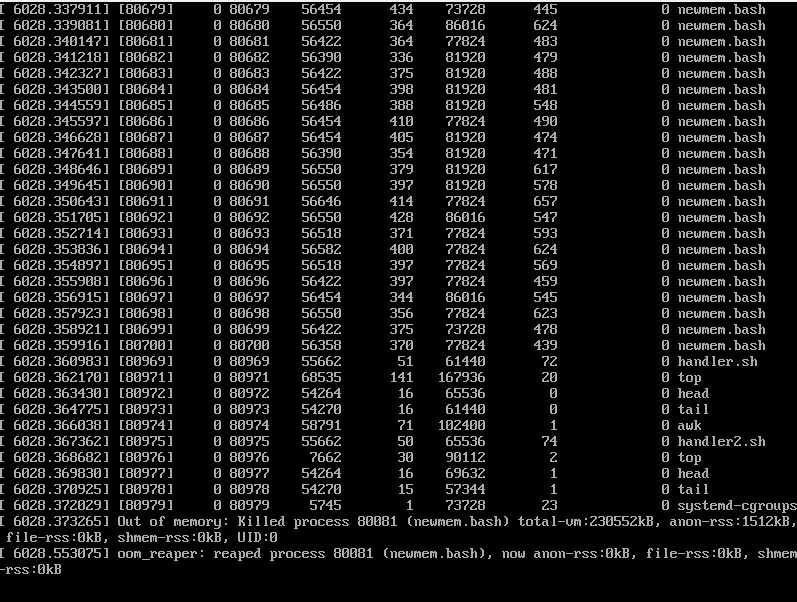
**Эксперимент 2**

Скопировал men.bash в файл newmem.bash и изменил условие в цикле. Также написал скрипт запуска newmen.bash 10 раз с ограничением по размеру массива на N, где N величина, в 10 раз меньшая, чем размер массива, при котором происходила аварийная остановка процесса в первом этапе предыдущего эксперимента.

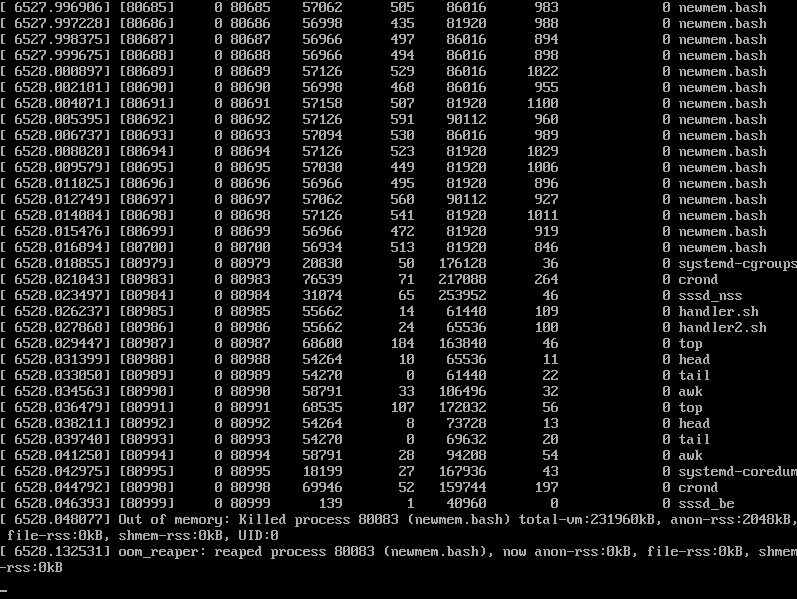
*Основной этап:*

Если устанавливать К = 10 и N = 1 860 000 программа успешно завершает свое выполнение, но

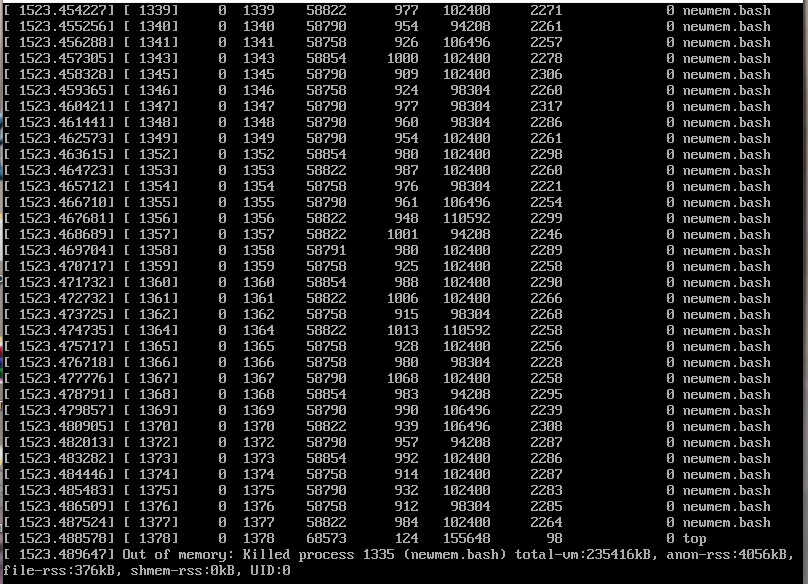
Если параметры равны К = 30 и N = 3000000 программа аварийно завершает свое выполнение.



Также если увеличить К в 3 раза и оставить N таким же, то программа снова аварийно завершит свое выполнение.



Теперь если N уменьшить до 1 860 000 , то программа успешно завершает свою работу.

Если увеличить на 140 000, то программа завершает аварийно выполнение, но если опустить на 10 000, то всё корректно. Следовательно программа продолжает работать корректно до чуть меньше 2 000 000, но когда доходит до 2, то завершает аварийно.

Как видно всё отработало корректно это, потому что процессы успевают закончиться раньше,

чем наступает критическое употребление памяти, когда завершаются процессы происходит

высвобождение памяти для следующих процессов.

Ряд процессов завершился аварийно, потому что при 30 запусках одновременно работают уже больше процессов, поэтому старые процессы не успевают завершиться. Следовательно память быстро расходуется и большинство процессов аварийно завершаются. Также, когда запускают новые процессы старые переходят в раздел подкачки, как наименее приоритетные и ожидают очереди на выполнение.

**Вывод**

Данные во время работы с оперативной памятью заполняют физическую память, пока не дойдут до аварийного состояния. Потом операционная система начинает заполнять файлы подкачки. Если файлов подкачки будет не хватать произойдет аварийная остановка процесса.