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

Выполнила студентка группы М3209 *Усольцева Валерия Игоревна*

*САНКТ-ПЕТЕРБУРГ*

*2020*

Отчет по Лабораторной работе №5

Конфигурация ОС:  
1. Объем оперативной памяти: 2 Гб  
2. Объем файла подкачки: 1 Гб  
3. Размер страницы памяти: 4 Кб  
4. Объем свободной оперативной памяти в ненагруженной системе: 120 Мб  
5. Объем свободного пространства в файле подкачки в ненагруженной системе: 1 Гб

Эксперимент №1  
Подготовительный этап:

Создадим скрипт mem.sh, добавляющий в конец массива 10 элементов и выводящий каждый 100000-й шаг размер массива в report.log

Создадим скрипт exec.sh, который запускает скрипт и считывает показания команды top каждую секунду.

Первый этап:

Запустим exec.sh, и подождем, пока система не убьет процесс:

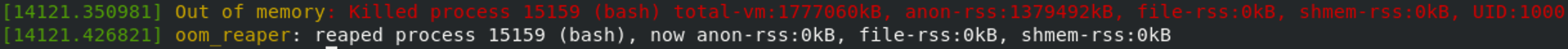
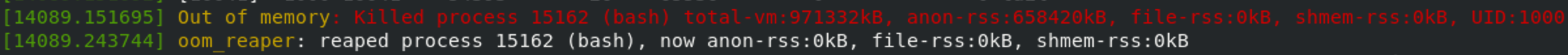
Зафиксируем последнюю строку системного отчета (так как dmesg | grep «mem.sh» не работает):

Зафиксируем последнюю строку файла report.log:

Построим график изменения размера оперативной памяти и размера файла подкачки от времени:

Второй этап:

Запустим exec1.sh и подождем, пока система не убьет процессы:

Зафиксируем строки системного отчета о завершении процессов:

Зафиксируем последние строки файлов report1.log:

И report2.log:

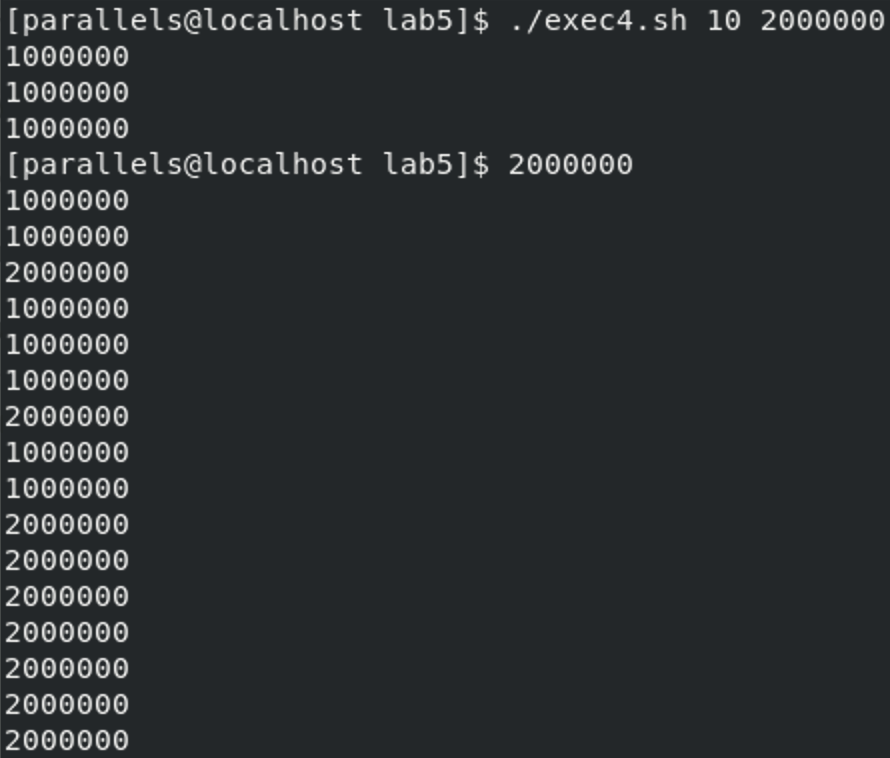
Построим график размера оперативной памяти и размера файла подкачки от времени:

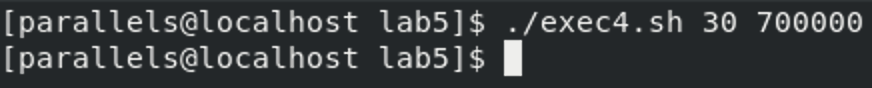
Наблюдения:

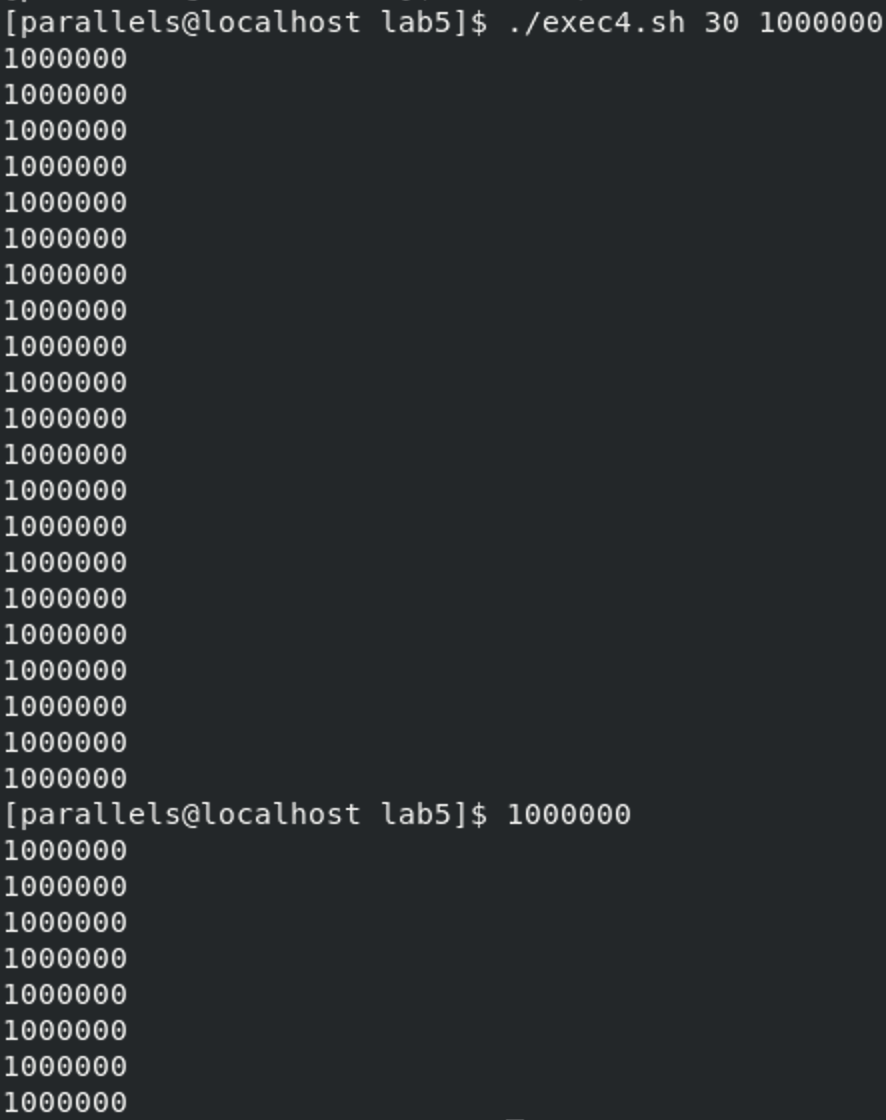
И в первом и во втором случаях размер массива ограничивается примерно 20000000 элементами. Когда оперативной памяти становится меньше 100 Мб, то данные начинают записываться в файл подкачки.

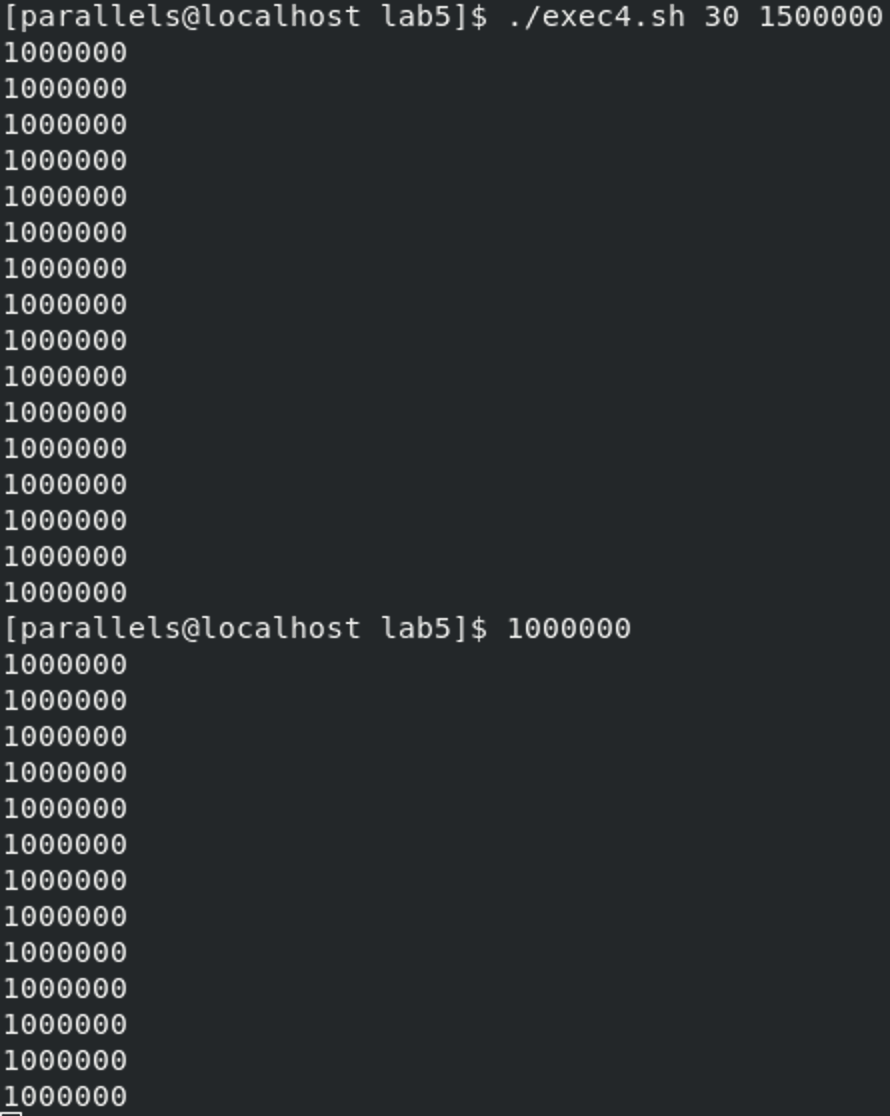
Скачок графика на 18 секунде можно объяснить тем, что в это время завершился процесс mem2.sh, освободив память. Также можно заметить, что во время остановки mem2.sh оперативной памяти освободилась половина, а файла подкачки освободилось меньше хотя успело заполниться чуть меньше половины (9000000 элементов) — это означает, что элементы из mem2.sh чаще записывались в оперативную память.

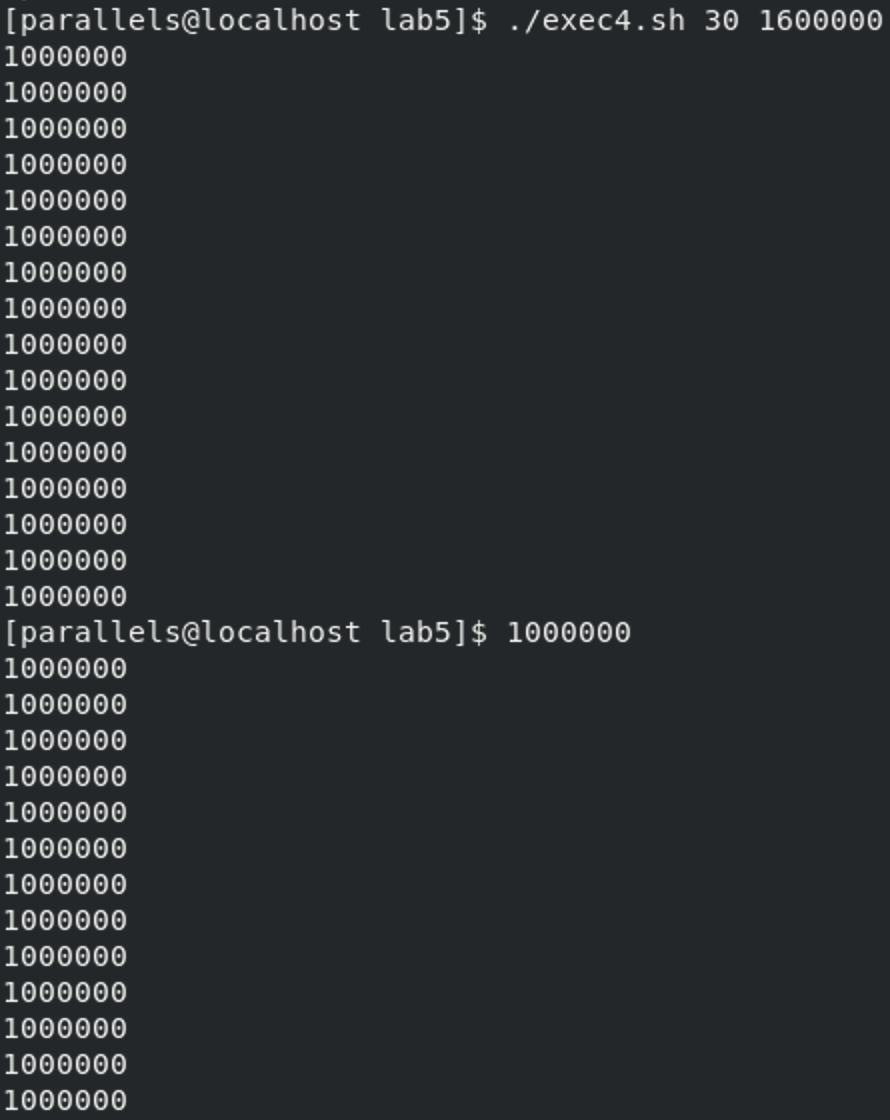
Эксперимент №2

Запустим exec4.sh 10 2000000, и увидим, что все скрипты отработали верно (подсчитаем количество 1000000 и 2000000 и увидим, что их одинаковое количество (10)):

После изменения на K=30 и запуске скриптов, у меня вылетает CentOS. Однако можно предположить, что при увеличении K скрипты, запущенные ранее, не успевают завершаться, освобождая память под следующие, тогда как при K=10 скрипты успевают отработать. Попробуем подобрать такое N, при котором скрипты не будут завершаться аварийно. Попробуем взять N равное в 3 раза меньше, чем брали для K=10 с округлением в большую сторону: . На этих значениях все скрипты завершаются без ошибок:

Возьмем тогда N=1000000. В этом случае все скрипты так же завершаются без ошибок:

В таком случае возьмем N=1500000. Опять же, все процессы завершились без аварии:

Возьмем N=1800000. В этом случае система снова вылетела.   
Возьмем N=1700000. В этом случае система снова вылетела.   
Возьмем N=1600000. В этом случае все процессы завершились без ошибок.

Значит, N=1600000 и есть искомое значение.

Наше N получилось не в 30 раз меньше потому, что процессы успевают отработать и освободить память.

Вывод: мы убедились на практике, что сначала заполняется оперативная память, при заполнении оперативной памяти начинает заполняться файл подкачки, а после заполнения файла подкачки, процесс аварийно завершается.