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 №M3204

***Триголос Алексей Павлович***

Проверила

***Титова Анастасия Витальевна***

***САНКТ-ПЕТЕРБУРГ***

***2020***

**Задание на лабораторную работу**

Проведите два виртуальных эксперимента в соответствии с требованиями и проанализируйте их результаты. В указаниях ниже описано, какие данные необходимо фиксировать в процессе проведения экспериментов.

Рекомендуется написать «следящие» скрипты и собирать данные, например, из вывода утилиты top автоматически с заданной периодичностью, например, 1 раз в секунду. Можно проводить эксперименты и фиксировать требуемые параметры и в ручном режиме, но в этом случае рекомендуется замедлить эксперимент, например, уменьшив размер добавляемой к массиву последовательности с 10 до 5 элементов.

**Текущая конфигурация операционной системы**

Общий объем оперативной памяти (MemTotal) = 1870644 кБ

Объем раздела подкачки (SwapTotal) = 839676 кБ

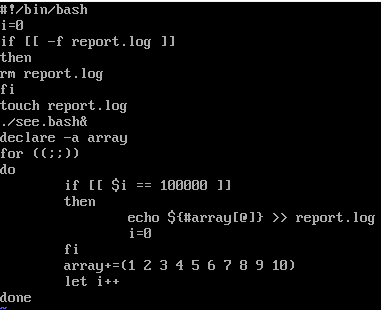
Размер страницы виртуальной памяти (Hugepagesize) = 2048 кБ

Объем свободной физической памяти в ненагруженной системе (MemFree) = 155492 кБ

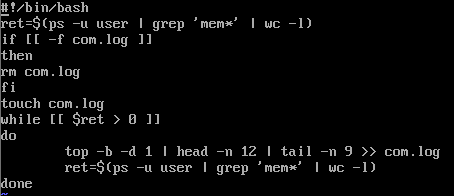
Объем свободного пространства в разделе подкачки в ненагруженной системе (SwapFree) = 750940 кБ

**Эксперимент 1**

mem.bash

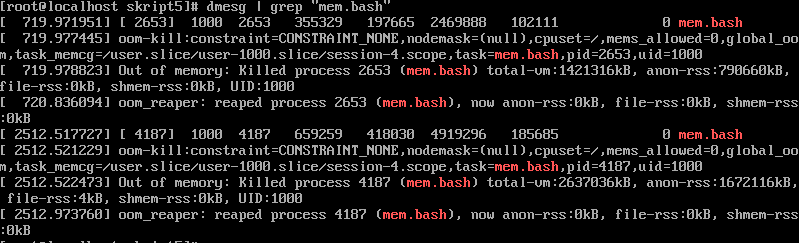


see.bash



*Первый этап:*

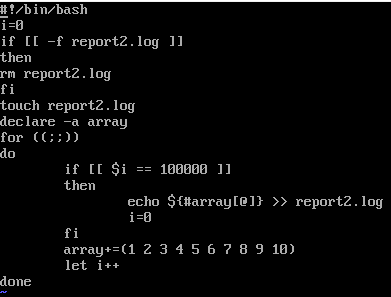
Последние записи о скрипте в журнале:



Последнее значение в report.log – **30000000**

*Второй этап:*

*mem2.bash*

**

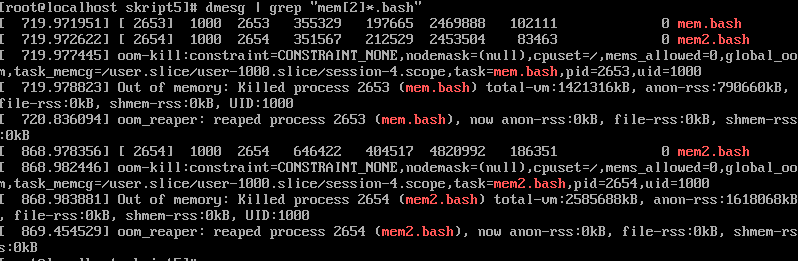
*gibrid.sh*

**

Последнее значение в report.log – **15000000**

Последнее значение в report2.log – **30000000**

Последние записи о скрипте в журнале:



**Графики по первому этапу первого эксперимента:**

**Графики по второму этапу первого эксперимента:**

**Изменения в верхних пяти процессах:**

Процессы, находящиеся в топ 5: mem.bash, mem2.bash, system, kthreadd, rcu\_gp, top, system-logind, kworker/0:1-events\_power\_ef+, kswapd0, ksoftirqd, dnf, kwoker/0:2-events.

Процессы, которые смогли попасть в топ 5 после удаления mem.bash: rcu\_par\_gp, kworker/0:1H-kblockd, kworker/0:0-events\_power\_ef+, kworker/0:0-mm\_percpu\_wq.

Обычно на самом верху были mem\*, top, system, kthreadd, rcu\_gp.

**Наблюдения по эксперименту 1:**

Первая часть:

Рассмотрим графики Mib Mem used и MiB Swap used. Можно, на первом из графиков происходит линейный процесс увеличения используемой памяти до определенного значения. При этом значении происходит переход на подкачку, пороговое изменение параметров и изменение наклона графика. После изменения наклона значения памяти не сильно отличаются друг от друга. Значит, можно сказать, что это почти линейный участок, но с другим наклоном.

На втором же графике мы видим обратную картину, сначала идет почти линейный участок, потом при том же значении времени, при котором менялся график Mib Mem used, меняется и Mib Swap used, но уже в линейный процесс.

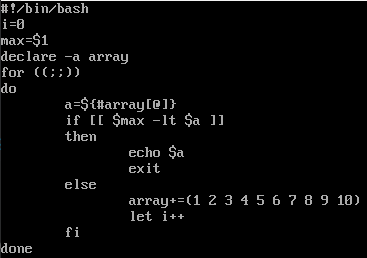
На графиках Mib Mem free и MiB Swap free. Видим, что Mem уменьшается почти линейно примерно до значения 60, Swap в это время не изменялся, но после этого, Mem практически не изменялся, а Swap примерно линейно уменьшался.

Вторая часть:

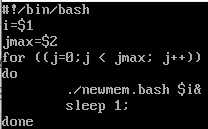
Во втором случае мы видим похожую картину, но только изменение наклона происходит трижды. Во второй раз аварийно выключается один из процессов, после чего происходит пороговое изменение параметров.

**Эксперимент 2**

*newmem.bash*

****

*script.sh*

****

Максимальное значение N при K = 30 – **1342179**

**Наблюдения во втором эксперименте:**

При **К = 10** и **N = 3000000** программа закончила свое выполнение без ошибок.

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

**Вывод:** в ходе экспериментов мы убедились, что массив сначала заполняет физическую память, но, когда память заканчивается, начинается выделение памяти из подкачки. Но при завершении память в разделе подкачки, процесс завершается аварийно. При запуске нескольких скриптов, если уменьшить максимальные размеры массивов в N раз и увеличить кол-во одинаковых процессов в N раз, то процессы завершатся без аварий.