Лабораторная работа №5

Отчет

* Общий̆ объем оперативной памяти = 1826.8 МБ
* Объем раздела подкачки = 820 МБ
* Размер страницы виртуальной памяти = 4096 Б
* Объем свободной физической памяти в ненагруженной системе = 1435 МБ
* Объем свободного пространства в разделе подкачки в ненагруженной системе = 820 МБ

1.1) Запустите созданный скрипт mem.bash. Дождитесь авариной остановки процесса и вывода в консоль последних сообщений системного журнала. Зафиксируете в отчете последнюю запись журнала - значения параметров, с которыми произошла аварийная остановка процесса.

Out of memory: Killed process 1723 (mem.bash) total-vm:2656964kB, anon-rss:1685000kB, file-rss:0kB, shmem-rss: 0kB, UID:0

oom\_reaper: reaped process 1723 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Killed

Также зафиксируйте значение в последней строке файла report.log.

3100\_0000

Подготовьте две консоли. В первой запустите утилиту top. Во второй запустите скрипт и переключитесь на первую консоль. Убедитесь, что в top появился запущенный̆ скрипт. Наблюдайте за следующими значениями (и фиксируйте их изменения во времени в отчете):

* значения параметров памяти системы (верхние две строки над основной̆ таблицей);
* значения параметров в строке таблицы, соответствующей̆ работающему скрипту;
* изменения в верхних пяти процессах (как меняется состав и позиции этих процессов).

Проводите наблюдения и фиксируйте их в отчете до авариной остановки процесса скрипта и его исчезновения из перечня процессов в top.

В файле top.logs

Посмотрите с помощью команды dmesg | grep "mem.bash" последние две записи о скрипте в системном журнале и зафиксируйте их в отчете.

Out of memory: Killed process 1755 (mem.bash) total-vm:2648252kB, anon-rss:1680648kB, file-rss:0kB, shmem-rss: 0kB, UID:0

oom\_reaper: reaped process 1755 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Killed

Также зафиксируйте значение в последней строке файла report.log.

31\_000\_000

1.2) Создайте копию скрипта, созданного на предыдущем этапе, в файл mem2.bash. Настройте её на запись в файл report2.log. Создайте скрипт, который запустит немедленно друг за другом оба скрипта в фоновом режиме. Подготовьте две консоли. В первой̆ запустите утилиту top. Во второй запустите созданный перед этим скрипт и переключитесь на первую консоль. Убедитесь, что в top появились mem.bash и mem2.bash. Наблюдайте за следующими значениями (и фиксируйте их изменения во времени в отчете):

* значения параметров памяти системы (верхние две строки над основной таблицей);
* значения параметров в строке таблицы, соответствующей работающему скрипту;
* изменения в верхних пяти процессах (как меняется состав и позиции этих процессов).

Проводите наблюдения и фиксируйте их в отчете до аварийной остановки последнего из двух скриптов и их исчезновения из перечня процессов в top.

В файле top2.logs

Посмотрите с помощью команды dmesg | grep "mem[2]\*.bash" последние записи о скриптах в системном журнале и зафиксируйте их в отчете.

Out of memory: Killed process 1538 (mem.bash) total-vm:1630268kB, anon-rss:991296 kB, file-rss:0kB, shmem-rss: 0kB, UID:0

oom\_reaper: reaped process 1538 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Killed

Out of memory: Killed process 1537 (mem.bash) total-vm:2657888kB, anon-rss:1690640kB, file-rss:0kB, shmem-rss: 0kB, UID:0

oom\_reaper: reaped process 1537 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Killed

Также зафиксируйте значения в последних строках файлов report.log и report2.log.

mem.bash: 31\_000\_000

mem2.bash: 18\_000\_000

Обработка результатов:

Постройте графики изменения каждой из величин, за которыми производилось наблюдение на каждом из этапов. Объясните динамику изменения этих величин исходя из теоретических основ управления памятью в рамках страничной организации памяти с разделом подкачки. Объясните значения пороговых величин: размер массива, при котором произошла аварийная остановка процесса, параметры, зафиксированные в момент аварийной остановки системным журналом. Сформулируйте письменные выводы.

Выводы:

Графики находятся в приложении 1.

На графике 1.1 видно, что 74% времени идет линейное уменьшение свободной оперативной памяти, при этом свап не изменяется. Когда количество свободной оперативной памяти перестает уменьшаться, оно начинает очень медленно расти вверх. При этом начинает использоваться раздел подкачки, который заканчивается примерно с линейной скоростью.

Когда начинает заканчиваться оперативная память, для выполнения процессов часто приходится использовать диск. Для этого используют виртуальную память (при виртуальной памяти - процесс может быть частично загружен в память для работы). В оперативной памяти всегда находится часть виртуального пространства, выделяемого для решения задачи, остальная его часть располагается на дисковой памяти. Если оперативной памяти не хватает для обеспечения работы текущего приложения, то приложение или его часть, которые не используют в данный момент процессор, выгружаются из оперативной памяти на диск. На их место в оперативную память загружается необходимый фрагмент активного приложения. Когда одному из выгруженных приложений передается управление, оно вновь загружается в оперативную память, что может привести к выгрузке на диск другого, пассивного в данный момент приложения. Таким образом, программы циркулируют между диском и оперативной памятью.

Поддержка виртуальной памяти позволяет открыть большое количество приложений

одновременно, но выгрузка на диск и загрузка с диска снижают производительность

компьютера. Используемая для этой цели часть внешней памяти называется файлом

подкачки. Процесс подкачки известен под названием свопинг. Объем файла подкачки может

в несколько раз превышать объем оперативной памяти.

Если посмотреть на график 1.2, то увидим, что возрастание использование оперативной памяти прекратилось на 20 итерации. Это значит, что %mem показывает, сколько оперативной памяти использует процесс, без учета использования свапа.

При просмотре графика 2.1 видим, что все аналогично, однако можно добавить, что оперативная память расходуется быстрее и ОС при нехватке памяти убивает процесс с максимальным потребление памяти.

Эксперимент №2   
Подготовительный этап:

Создайте копию скрипта mem.bash в файл newmem.bash. Измените копию таким образом, чтобы она завершала работу, как только размер создаваемого массива превысит значение N, передаваемое в качестве параметра скрипту. Уберите запись данных в файл.

Основной этап:

Задача – определить граничные значения потребления памяти, обеспечивающие безаварийную работу для регулярных процессов, запускающихся с заданной интенсивностью.  
Ход эксперимента:  
Создайте скрипт, который будет запускать newmem.bash каждую секунду, используя один и тот же параметр N так, что всего будет осуществлено K запусков.

Возьмите в качестве значения N, величину, в 10 раз меньшую, чем размер массива, при котором происходила аварийная остановка процесса в первом этапе предыдущего эксперимента. Возьмите в качестве K значение 10. Убедитесь, что все K запусков успешно завершились, и в системном журнале нет записей об аварийной остановке newmem.bash.

Измените значение K на 30 и снова запустите скрипт. Объясните, почему ряд процессов завершился аварийно. Подберите такое максимальное значение N, чтобы при K=30 не происходило аварийных завершений процессов. Укажите в отчете сформулированные выводы по этому эксперименту и найденное значение N.

Выводы:

По результатам второго эксперимента с помощью алгоритма бинарного поиска было получено, что граничное значение размера массива, выделенного на процесс примерно 1\_600\_000 (общий размер примерно 48\_000\_000)

Начальное N = 31\_000\_000

Общий размер увеличился, так как при запуске 30 скриптов, процессор распределяется на процессы не совсем равномерно и в определенных процессах идет достижение нужного размера массива быстрее, чем в других. Когда процесс завершает работу, он отдает память, задействованную им, другим процессам. Так же стоит упомянуть, что процессы запускается с периодичностью в 1 секунду. И тем самым удалось достичь такого результата.

Приложение 1.

1.1)

1.2)

2.1)

2.2)