Лабораторная работа V

Управление памятью в ОС Linux

Методические материалы к заданию

Основные источники данных о состоянии памяти вычислительного узла

- файл `/proc/meminfo` (документация в соответствующем разделе `man proc`);
- файл `/proc/[PID]/statm` (документация в соответствующем разделе `man proc`);
- утилита `top`;

Организация управления памятью в ОС Linux

В Linux используется страничная организация виртуальной памяти. Память разбита на страницы. Размер страницы можно посмотреть в параметрах конфигурации с помощью команды **`getconf PAGE_SIZE`**. При обращении к адресу в памяти происходит динамическое преобразование адреса путем замены старших бит виртуального адреса на номер физической страницы с сохранением значения младших бит как смещения на странице. Обычно, кроме физической памяти используется также раздел подкачки. В этом случае

адресное пространство процесса состоит из страниц, находящихся в оперативной памяти и страниц, находящихся в разделе подкачки. Параметры раздела подкачки можно узнать из файла `/proc/swaps`. При динамическом выделении памяти процессу, операционная система сначала пытается выделить страницы в физической памяти, но если это невозможно, инициирует страничный обмен, в рамках которого ряд страниц из физической памяти вытесняется на раздел подкачки, а адреса, соответствующие вытесненным страницам выделяются процессу под новые страницы. Операционная система контролирует выделение памяти процессам. Если процесс попробует запросить расширение адресного пространства, которое невозможно в пределах имеющейся

Задание

Требуется провести два виртуальных эксперимента в соответствии с требованиями и проанализировать 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.

Требования к проведению экспериментов и содержанию отчета

1. Перед началом экспериментов зафиксируйте в отчете данные о текущей конфигурации операционной системы в аспекте управления памятью:
   - Общий объем оперативной памяти;
   - Объем раздела подкачки;
   - Размер страницы виртуальной памяти;
   - Объем свободной физической памяти в ненагруженной системе;
   - Объем свободного пространства в разделе подкачки в ненагруженной системе;
2. Минимальные требования к отчету
   - конфигурация из пунтка выше
   - значения файлов `.log`
   - графики
   - ваши наблюдения и выводы
   - человекочитаемое оформление

   Отчет можно делать в любом удобном вам формате: PDF-файл, Jupyter Notebook и т.п.

Эксперимент №1

Подготовительный этап:

Создайте скрипт **`mem.bash`**, реализующий следующий сценарий:

1. Создается пустой массив и счетчик шагов.
2. Выполняется бесконечный цикл, на каждом шаге которого в конец массива добавляется последовательность из 10 элементов, например, (1 2 3 4 5 6 7 8 9 10).
3. Каждый 100000-ый шаг в файл report.log добавляется строка с текущим значением размера массива (перед запуском скрипта, файл обнуляется).

Первый этап:

Запустите созданный скрипт **`mem.bash`**. Дождитесь аварийной остановки процесса и вывода в консоль последних сообщений системного журнала. Зафиксируйте в отчете последнюю

запись журнала - значения параметров, с которыми произошла аварийная остановка процесса. Также зафиксируйте значение в последней строке файла **report.log**.

Наблюдение:

Подготовьте две консоли. В первой запустите утилиту **top**. Во второй запустите скрипт и переключитесь на первую консоль. Убедитесь, что в **top** появился запущенный скрипт. 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 процесса скрипта и его исчезновения из перечня процессов в **top**. Посмотрите с помощью команды **dmesg | grep "mem.bash**“ последние две записи о скрипте в системном журнале и зафиксируйте их в отчете. Также зафиксируйте значение в последней строке файла **report.log**.

Второй этап:

Создайте копию скрипта, созданного на предыдущем этапе, в файл **mem2.bash**. Настройте ее на запись в файл **report2.log**. Создайте скрипт, который запустит оба скрипта в фоновом режиме одновременно.

Проведите аналогичное *наблюдение* в процессе работы двух экземпляров созданного скрипта (**mem.bash** и **mem2.bash**).

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.

Эксперимент №2

Подготовительный этап:

Скопируйте **mem.bash** в новый файл – **newmem.bash**: новый скрипт должен завершать работу, как только размер создаваемого массива превысит значение **N**, которое передается в качестве параметра. Также уберите в **newmem.bash** запись в файл.

Основной этап:

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**.