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

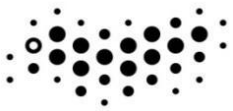
Отчет

По лабораторной работе «Управление памятью в ОС Linux»

По дисциплине «Операционные Системы»

Выполнил студент группы №M3203
Кулешова Екатерина Дмитриевна

Преподаватели
Маятин Александр Владимирович
Титова Анастасия Витальевна



УНИВЕРСИТЕТ ИТМО

САНКТ-ПЕТЕРБУРГ

2020

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Задачи работы

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.

Данные о текущей конфигурации системы

Скрипт для сбора информации

```
#!/bin/bash

#system config

> config
awk '$1 == "MemTotal:" {print "The total amount of RAM: " $2 " " $3}' /proc/meminfo >> config
awk '$1 == "SwapTotal:" {print "The size of the swap partition: " $2 " " $3}' /proc/meminfo >> config
echo "Virtual memory page size: $(getconf PAGE_SIZE) B" >> config
awk '$1 == "MemFree:" {print "The amount of free RAM on an unloaded system: " $2 " " $3}' /proc/meminfo >> config
awk '$1 == "SwapFree:" {print "The amount of free space in the swap partition on an unloaded system: " $2 " " $3}' /proc/meminfo >> config
```

Данные

```
The total amount of RAM: 1870900 kB
The size of the swap partition: 839676 kB
Virtual memory page size: 4096 B
The amount of free RAM on an unloaded system: 1709940 kB
The amount of free space in the swap partition on an unloaded system: 751364 kB
```

ЭКСПЕРИМЕНТ 1

Подготовительный этап:

Создайте скрипт mem.bash, реализующий следующий сценарий.

Скрипт выполняет бесконечный цикл. Перед началом выполнения цикла создается пустой массив и счетчик шагов, инициализированный нулем. На каждом шаге цикла в конец массива добавляется последовательность из 10 элементов, например, (1 2 3 4 5 6 7 8 9 10).

Каждый 100000-ый шаг в файл report.log добавляется строка с текущим значением размера массива (перед запуском скрипта, файл обнуляется).

Первый этап:

Задача

– оценить изменения параметров, выводимых утилитой top в процессе работы созданного скрипта.

Ход эксперимента

- 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

```
[ 1891.824905] [ 1483]  1000  1483   664967   416131  4952064   193263             0 mem.bash
[ 1891.826683] Out of memory: Killed process 1483 (mem.bash) total-vm:2659868kB, anon-rss:1664524kB,
 file-rss:0kB, shmem-rss:0kB, UID:1000
[ 1892.146818] oom_reaper: reaped process 1483 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss
:0kB
Killed
[user@localhost experiment1]$
```

- Также зафиксируйте значение в последней строке файла report.log.

```
[user@localhost experiment1]$ cat report.log | tail -n 1
31000000
```

СЛЕЖЕНИЕ 1

Скрипт tracker.sh (выполняет все слежения)

```
#!/bin/bash

if [[ $# -ne 3 ]]
then
        echo "Three parameters expected: report_file, ram_file, swap_file to added information about experiment"
        exit
fi

report_file=$1
ram_file=$2
swap_file=$3

> "$report_file"
> "$ram_file"
> "$swap_file"

while [[ true ]]
do
        top -b -n 1 > cur_top
        skript_info=$(cat cur_top | grep "mem[2]*.bash" | awk '$8 == "R" {print $0}')

        if [[ -n "$skript_info" ]]
        then
                cur_time=$(date +%X)

                echo "Time: $cur_time">> "$report_file"

                #memory info
                echo "Informattion about memory:" >> "$report_file"
                awk '$1 == "MiB" || $1 == "PID" {print $0}' cur_top >> "$report_file"
                awk '$2 == "Mem" {print $3 " " $cur_time}' cur_top >> "$ram_file"
                awk '$2 == "Swap" {print $3 " " $cur_time}' cur_top >> "$swap_file"

                #skript info
                echo "Information about mem.bash:" >> "$report_file"
                echo "$skript_info" >> "$report_file"

                  #first five processes
                  echo "Information about first five processes:" >> "$report_file"
                  cat cur_top | head -n 12 | tail -n 6 >> "$report_file"

                  echo >> "$report_file"
                  rm cur_top
         else
                  rm cur_top
                  echo "Last info in dmesg: " >> "$report_file"
                  dmesg | grep "mem.bash" | tail -n 2 >> "$report_file"
                  exit
         fi
         sleep 1
done
```

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.

Представленный скрипт осуществляет слежение за работой mem.bash и фиксирует все запуски вплоть до нехватки памяти

Часть логирующего файла report1.txt

Конец файла с информацией из dmesg:

```
Time: 11:04:23 AM
Informattion about memory:
MiB Mem :   1827.1 total,     73.6 free,   1725.9 used,     27.6 buff/cache
MiB Swap:    820.0 total,      0.0 free,    820.0 used.     13.8 avail Mem
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
Information about mem.bash:
  10270 user      20   0 1516880   1.2g    548 R  38.6  69.2   1:38.11 mem.bash
Information about first five processes:
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  10270 user      20   0 1516880   1.2g    548 R  38.6  69.2   1:38.11 mem.bash
     49 root      20   0       0      0      0 S   4.3   0.0   9:03.75 kswapd0
  11323 user      20   0  274140   1720   1004 R   2.9   0.1   0:00.04 top
      1 root      20   0  179196   1980      0 S   0.0   0.1   0:05.50 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd

Last info in dmesg:
[11212.472620] [10270]  1000 10270   383840   328274  2699264        0             0 mem.bash
[11212.478728] Out of memory: Killed process 10270 (mem.bash) total-vm:1535360kB, anon-rss:1313096kB, file-rss:0kB, shmem-rss:0kB, UID:1000
```

Начало с общей структурой:

```
Time: 11:01:33 AM
Information about memory:
MiB Mem :   1827.1 total,   1257.3 free,    492.4 used,     77.4 buff/cache
MiB Swap:    820.0 total,      0.2 free,    819.8 used.   1222.4 avail Mem
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
Information about mem.bash:
  10270 user      20   0  255092  35740   3004 R  94.1   1.9   0:02.40 mem.bash
Information about first five processes:
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  10270 user      20   0  255092  35740   3004 R  94.1   1.9   0:02.40 mem.bash
      1 root      20   0  179196   2548    568 S   0.0   0.1   0:05.50 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp

Time: 11:01:34 AM
Information about memory:
MiB Mem :   1827.1 total,   1239.9 free,    509.6 used,     77.6 buff/cache
MiB Swap:    820.0 total,      0.2 free,    819.8 used.   1205.1 avail Mem
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
Information about mem.bash:
  10270 user      20   0  272648  53428   3004 R  93.8   2.9   0:03.68 mem.bash
Information about first five processes:
    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
  10270 user      20   0  272648  53428   3004 R  93.8   2.9   0:03.68 mem.bash
      1 root      20   0  179196   2548    568 S   0.0   0.1   0:05.50 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
```

По данным видно, что сначала расходуется оперативная память, когда она заканчивается, происходит подкачка из swap. За счет этого в самом конце раздел подкачки не имеет свободного места, но оперативная память за один процесс до аварийной остановки - да.

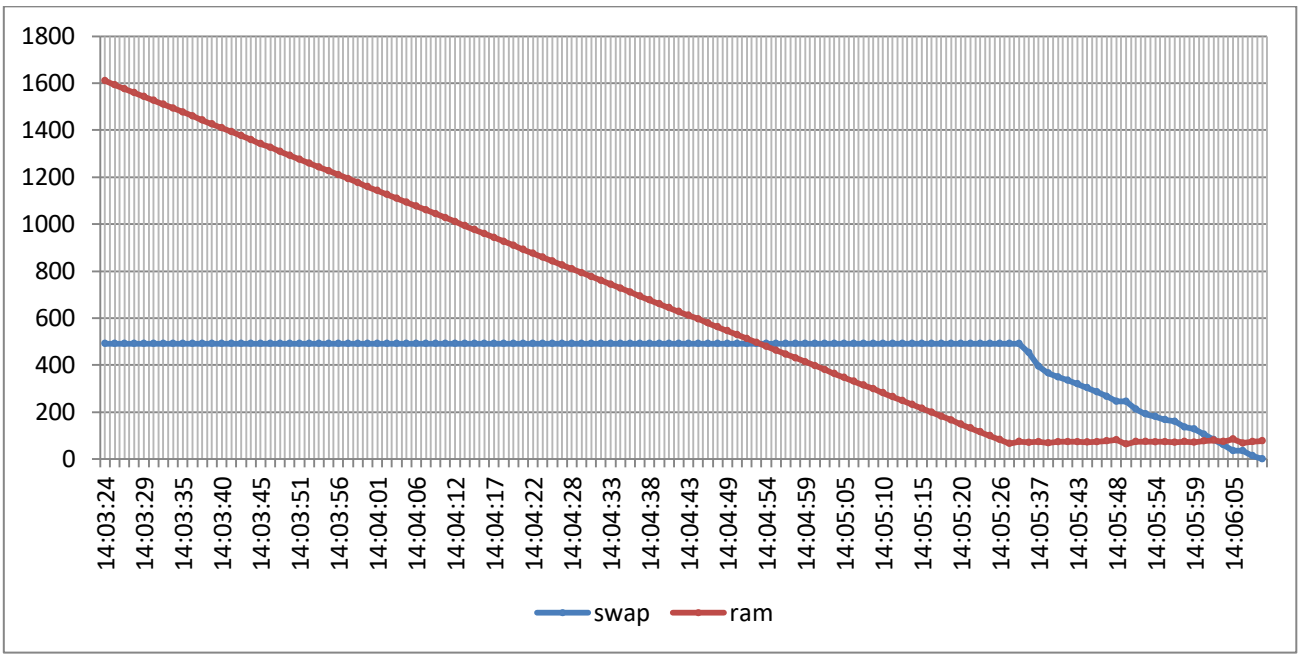
Последняя строка в файле report.log:

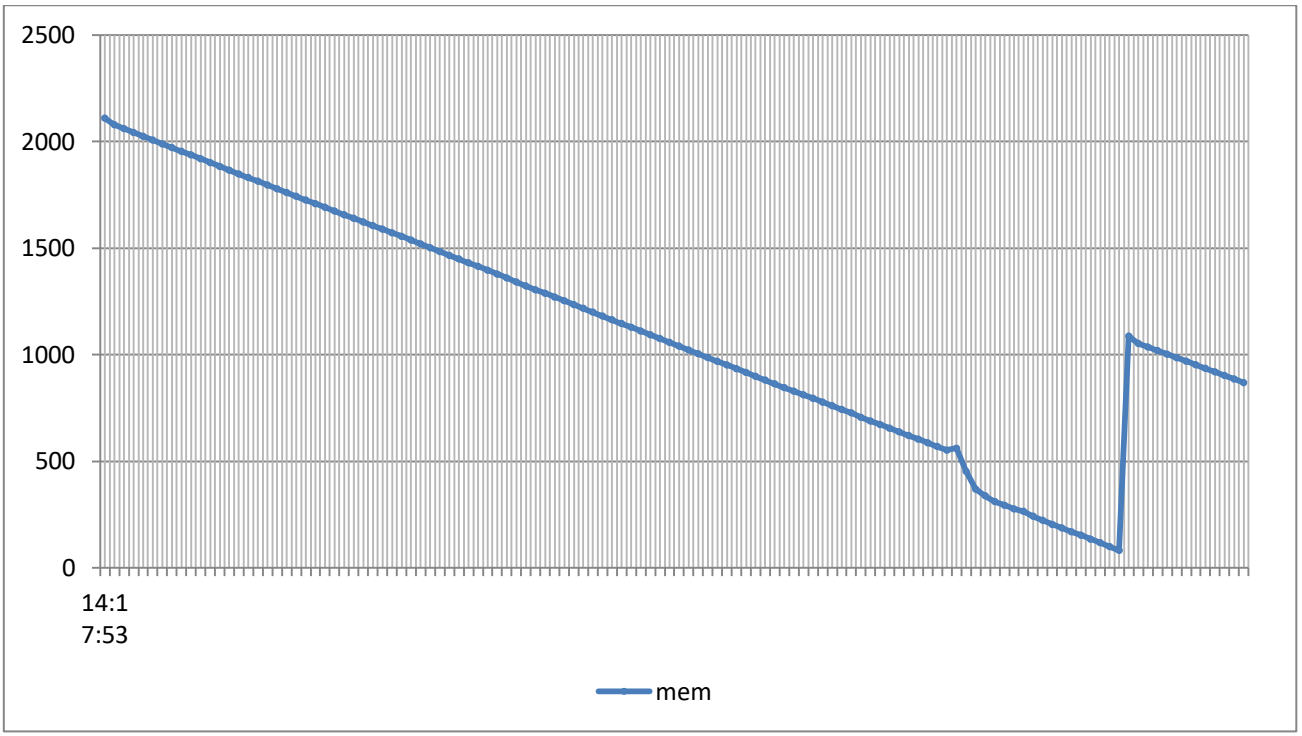
```
16000000
```

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.

Графики:


1800
1600
1400
1200
1000
800
600
400
200
0
14:03:24
14:03:29
14:03:35
14:03:40
14:03:45
14:03:51
14:03:56
14:04:01
14:04:06
14:04:12
14:04:17
14:04:22
14:04:28
14:04:33
14:04:38
14:04:43
14:04:49
14:04:54
14:04:59
14:05:05
14:05:10
14:05:15
14:05:20
14:05:26
14:05:37
14:05:43
14:05:48
14:05:54
14:05:59
14:06:05
swap
ram


2500
2000
1500
1000
500
0
14:1
7:53
mem

СЛЕЖЕНИЕ 2

Задача

– оценить изменения параметров, выводимых утилитой top в процессе работы нескольких экземпляров созданного скрипта.

Ход эксперимента:

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 первой 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;

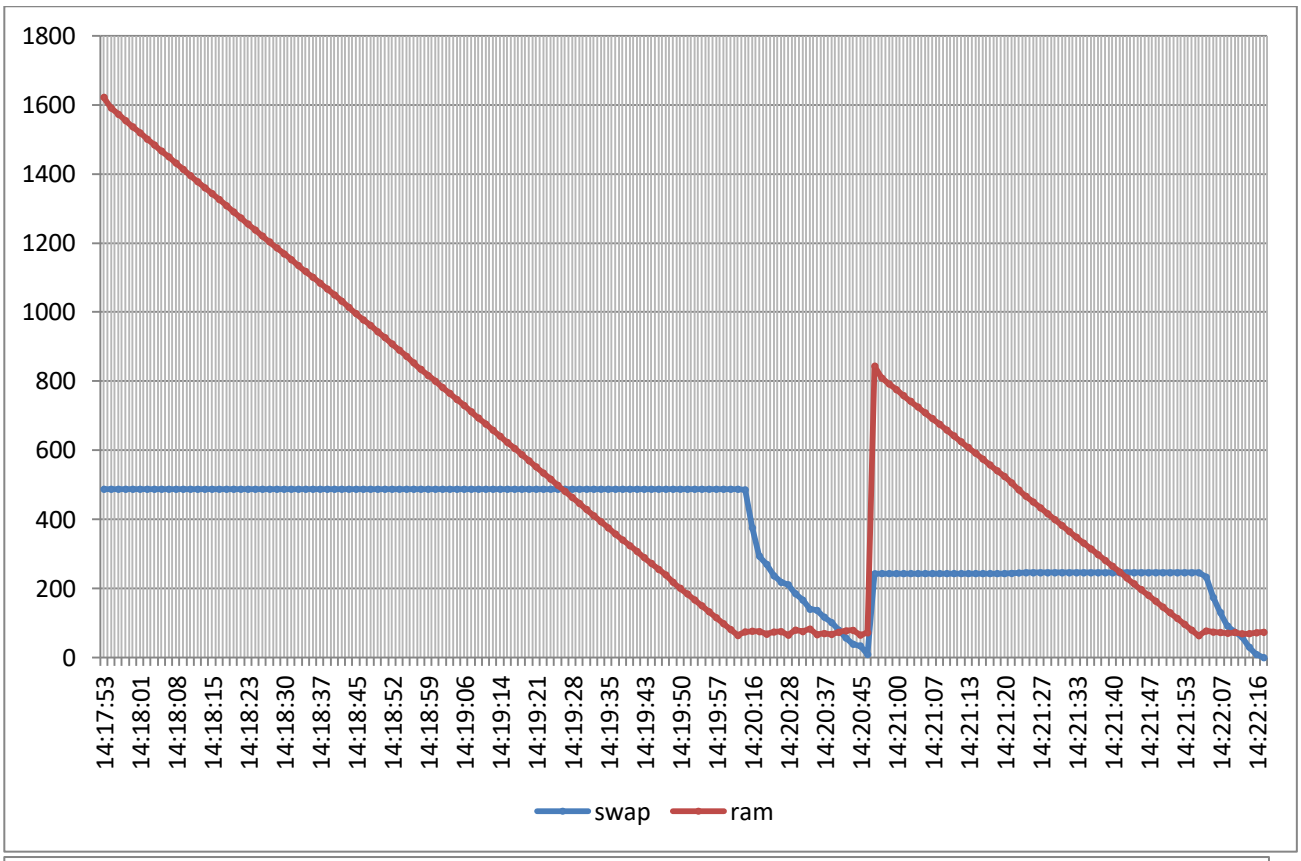
 изменения в верхних пяти процессах (как меняется состав и позиции этих процессов).

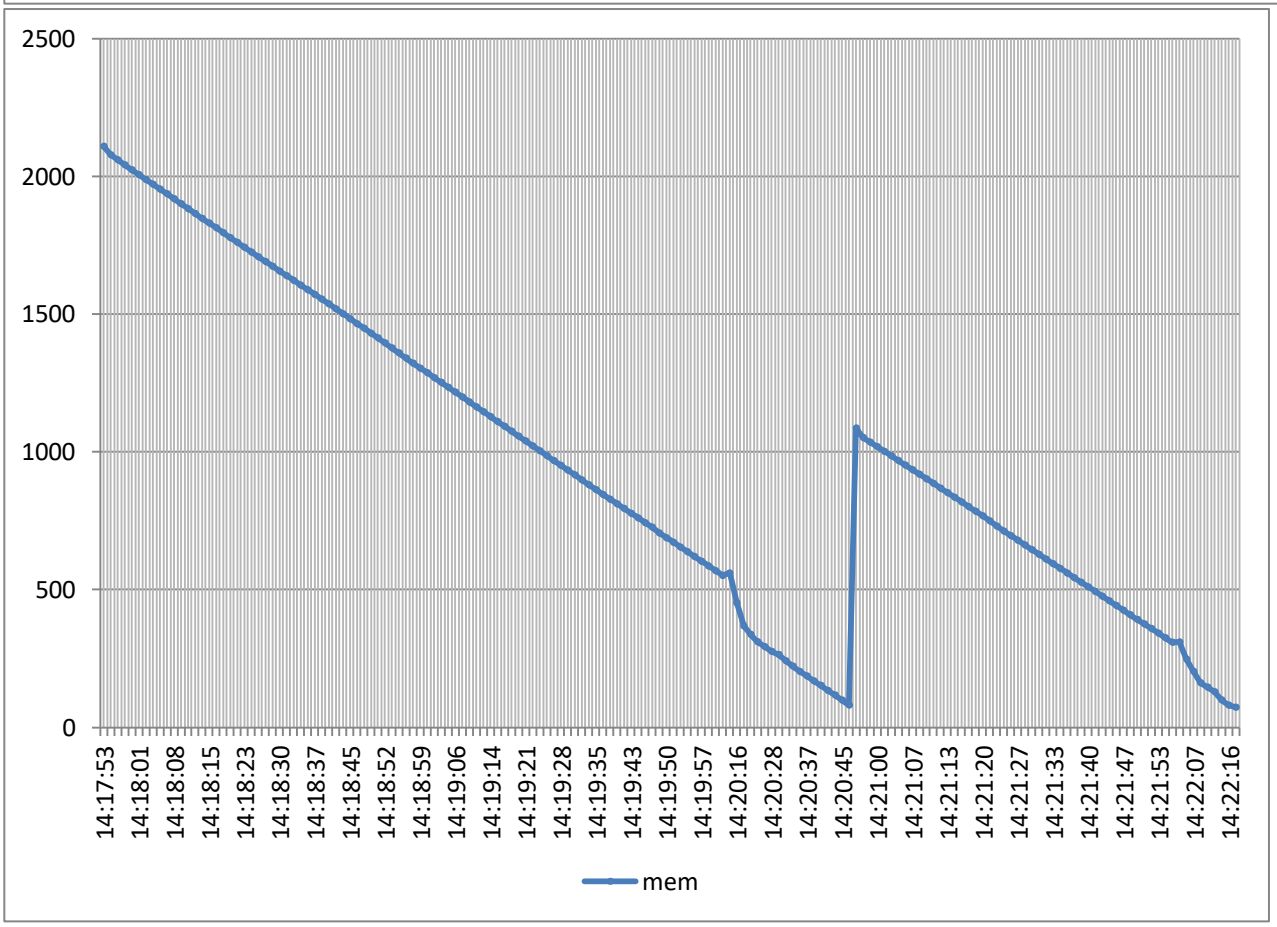
Проводите наблюдения и фиксируйте их в отчете до аварийной остановки последнего из двух скриптов и их исчезновения из перечня процессов в top.

Посмотрите с помощью команды dmesg | grep "mem[2]*.bash" последние записи о скриптах в системном журнале и зафиксируйте их в отчете. Также зафиксируйте значения в последних строках файлов report1.log и report2.log.

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.


1800
1600
1400
1200
1000
800
600
400
200
0
14:17:53
14:18:01
14:18:08
14:18:15
14:18:23
14:18:30
14:18:37
14:18:45
14:18:52
14:18:59
14:19:06
14:19:14
14:19:21
14:19:28
14:19:35
14:19:43
14:19:50
14:19:57
14:20:16
14:20:28
14:20:37
14:20:45
14:21:00
14:21:07
14:21:13
14:21:20
14:21:27
14:21:33
14:21:40
14:21:47
14:21:53
14:22:07
14:22:16
swap
ram


2500
2000
1500
1000
500
0
14:17:53
14:18:01
14:18:08
14:18:15
14:18:23
14:18:30
14:18:37
14:18:45
14:18:52
14:18:59
14:19:06
14:19:14
14:19:21
14:19:28
14:19:35
14:19:43
14:19:50
14:19:57
14:20:16
14:20:28
14:20:37
14:20:45
14:21:00
14:21:07
14:21:13
14:21:20
14:21:27
14:21:33
14:21:40
14:21:47
14:21:53
14:22:07
14:22:16
mem

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.

```
#!/bin/bash

if [[ $# -ne 1 ]]
then
        echo "One parameter needed: array size"
        exit
fi

arr=()
step=0

while [[ true ]]
do
        if [[ ${#arr[@]} -gt $1 ]]
        then
                exit
        fi

        (( step++ ))
        arr+=( 1 2 3 4 5 6 7 8 9 9 )
done
```

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.

```
#!/bin/bash

if [[ $# -ne 2 ]]
then
        echo "Two parameters needed: count, array size"
        exit
fi

for (( i=0; i<$2; i++ ))
do
        sleep 1
        ./newmem.bash $1 &
done
```

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.

Из первого эксперимента знаем критический размер массива. Последняя строка в файле report.log:

```
16000000
```

Убедитесь, что все K запусков успешно завершились, и в системном журнале нет записей об аварийной остановке newmem.bash.

```
[user@localhost experiment2]$ ./tester.sh 1600000 10
[user@localhost experiment2]$
```

Все к запусков успешно отработали. Это обуславливается тем, что процессы успевают закончиться раньше, чем наступает критическое употребление памяти. По мере завершения процессов происходит высвобождение памяти для следующих процессов. Все выполняется параллельно и успешно

Измените значение K на 30 и снова запустите скрипт. Объясните, почему ряд процессов завершился аварийно.

При 30 запусках одновременно работают уже больше процессов. Старые не успевают завершиться, как инициируются новые. Память быстро расходуется. Как результат, многие процессы аварийно завершаются. Чем освобождают физическую память для других процессов. Также, при запуске новых процессов старые переходят в раздел подкачки как менее приоритетные и ждут своей очереди на выполнение. Данный процесс также не бесконечен. Размер буфера подкачки ограничен, и выделяется для помощи оперативной памяти

Подберите такое максимальное значение N, чтобы при K=30 не происходило аварийных завершений процессов.

Для N= 1300 000 все работает корректно

Для N=1400 000, 1 450 000 также корректно

Однако, на значении N=1 460 000 происходить kill некоторых процессов

```
[ 7347.298767] [16183]     0 16183    75752    16032   233472     4285             0 newmem.bash
[ 7347.299984] [16185]     0 16185    75818    16282   233472     4083             0 newmem.bash
[ 7347.301225] Out of memory: Killed process 16134 (newmem.bash) total-vm:335216kB, anon-rss:64196kB
, file-rss:572kB, shmem-rss:0kB, UID:0
[ 7350.064838] oom_reaper: reaped process 16134 (newmem.bash), now anon-rss:0kB, file-rss:0kB, shmem
-rss:0kB
```

Значит, оптимальное значение находится в промежутке [1 450 000; 1 460 000)

```
[root@localhost experiment2]# ./tester.sh 1450000 30
[root@localhost experiment2]#
```

Укажите в отчете сформулированные выводы по этому эксперименту и найденное значение N.

ВЫВОДЫ

В результате экспериментов можно сделать вывод о том, как работает память в Linux

Сначала расходуется физическая память для всех процессов. Как только достигается значение, необходимое системе для работы, начинается подкачка памяти из раздета swap. Этот процесс происходит либо до тех пор, пока не закончится вся память, тогда некоторые процессы аварийно завершатся. Освободив тем саамы оперативную память. Некоторые разделы вернуться в блок подкачки

ИСТОЧНИКИ

1. Методичка с заданием к лабораторной работе[1]

2. Открытая документация bash[2]

[1] https://drive.google.com/file/d/1N9JZKzWwMzpC8QdZ0-mY-ib9qouAy_Ua/view
[2] http://www.opennet.ru/man.shtml?topic=&russian=0&category=&submit=%F0%CF%CB%C1%DA%C1%D4%D8+man