Отчет по лабораторной работе 5

Данные о текущей конфигурации операционной системы в аспекте управления памятью:

- Общий объем оперативной памяти: 4005120 kB;
- Объем раздела подкачки: 4004860 kB;
- Размер страницы виртуальной памяти: 4 kB;
- Объем свободной физической памяти в ненагруженной системе: 2563716 kB;
- Объем свободного пространства в разделе подкачки в ненагруженной системе: 4004860 kB.

Эксперимент №1

Первый этап

Запуск **mem.sh** привел к аварийной остановке процесса.
Последняя запись в журнале, полученная с помощью **dmesg | grep "mem.sh"**:

```
[15822.955087] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=user
.slice,mems_allowed=0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000
.service/app.slice/app-org.gnome.Terminal.slice/vte-spawn-c8a7aabf-6f63-4cf2-8cbe-
3b68d4e29e69.scope,task=mem.sh,pid=24811,uid=1000
[15822.955096] Out of memory: Killed process 24811 (mem.sh) total-vm:6995356kB,
anon-rss:3545648kB, file-rss:1680kB, shmem-rss:0kB, UID:1000 pgtables:13720kB
oom_score_adj:0
```

Значение в последней строке **report.log**: 89000000.
Во время работы **mem.sh** необходимые данные утилиты **top** фиксировались с помощью скрипта **monitor1.sh** в файл **.mem_data** ежесекундно.
Результаты наблюдений за оперативной памятью зафиксированы на Рис.1. На протяжении всей работы **mem.sh** скрипт находился на первом месте утилиты **top** по потреблению памяти.

Второй этап

Запуск **mem.sh** и **mem2.sh** сначала привел к аварийной остановке процесса **mem.sh**, а затем и к аварийной остановке процесса **mem2.sh**.
Последняя запись в журнале, полученная с помощью **dmesg | grep "mem[2]*.sh"**:

```
[25655.869335] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=user
.slice,mems_allowed=0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000
```

```
.service/app.slice/app-org.gnome.Terminal.slice/vte-spawn-7d0bf107-3c62-4fd9-bd1c-
9909f9ab40af.scope,task=mem.sh,pid=27327,uid=1000
[25655.869345] Out of memory: Killed process 27327 (mem.sh) total-vm:3501316kB,
anon-rss:1780200kB, file-rss:1616kB, shmem-rss:0kB, UID:1000 pgtables:6888kB
oom_score_adj:0
[25701.110312] [ 27328] 1000 27328 1750060 898948 14061568 849931 0 mem2.sh
[25701.110316] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=user
.slice,mems_allowed=0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000
.service/app.slice/app-org.gnome.Terminal.slice/vte-spawn-7d0bf107-3c62-4fd9-bd1c-
9909f9ab40af.scope,task=mem2.sh,pid=27328,uid=1000
[25701.110327] Out of memory: Killed process 27328 (mem2.sh) total-vm:7000240kB,
anon-rss:3593960kB, file-rss:1832kB, shmem-rss:0kB, UID:1000 pgtables:13732kB
oom_score_adj:0
```

Значение в последней строке **report.log**: 44000000.
Значение в последней строке **report2.log**: 89000000.
Во время работы **mem.sh** и **mem2.sh** необходимые данные утилиты **top** фиксировались с помощью скрипта **monitor2.sh** в файл **.mem2_data** ежесекундно.
Результаты наблюдений за оперативной памятью зафиксированы на Рис.2. До аварийной остановки **mem.sh** скрипты делили первое место утилиты **top** по потреблению памяти. Потом **mem2.sh** занимал первое место до самого конца.

Вывод

1. Из первого этапа видно, что при полном заполнении физической памяти (отмечена синим на Рис.1) начинается использование раздела подкачки, то есть происходит страничный обмен и его заполнение. При полном заполнении файла подкачки произошла аварийная остановка процесса.

2. Из второго этапа видно, что оба скрипта аналогично заполнили всю физическую память и файл подкачки. Для продолжения работы хотя бы одного процесса был аварийно остановлен **mem.sh**. Поэтому в середине графика можно заметить, как резко освободилась вся память, затраченная **mem.sh**. Затем **mem2.sh** заполнил всю освобожденную физическую память и файл подкачки, а потом аварийно завершился.

Эксперимент №2

Основной этап и вывод

Запуск **newmem.sh** с аргументом $N = 8900000$ и $K = 10$ завершился успешно. После выполнения **dmesg | grep "newmem.sh"** ничего не было выведено. Это можно объяснить тем, что в худшем случае общий размер всех массивов был равен $\frac{N_{total}}{K} \cdot K = N_{total} = 89000000$.

Запуск **newmem.sh** с аргументом $N = 8900000$ и $K = 30$ завершился аварийно. После выполнения **dmesg | grep "newmem.sh"** вывелось 15 аварийных сообщениий. Очевидно, что даже в лучшем случае общий размер всех массивов будет больше, чем $N_{total} = 89000000$.

Подобрать приближенное максимальное значение N при $K = 30$ без аварийной остановки можно с помощью бинарного поиска. Сделаем левой границей $l = \frac{N_{total}}{K} = 2966666$, а правой границей $r = N = 8900000$. После выполнений нескольких итераций приходим к примерному максимальному значению $N = 5400000$, при котором выполение **dmesg | grep "newmem.sh"** ничего не выводит. Стоит уточнить, что итерации выполнялись вручную, так как для завершения всех процессов необходимо некоторое время.

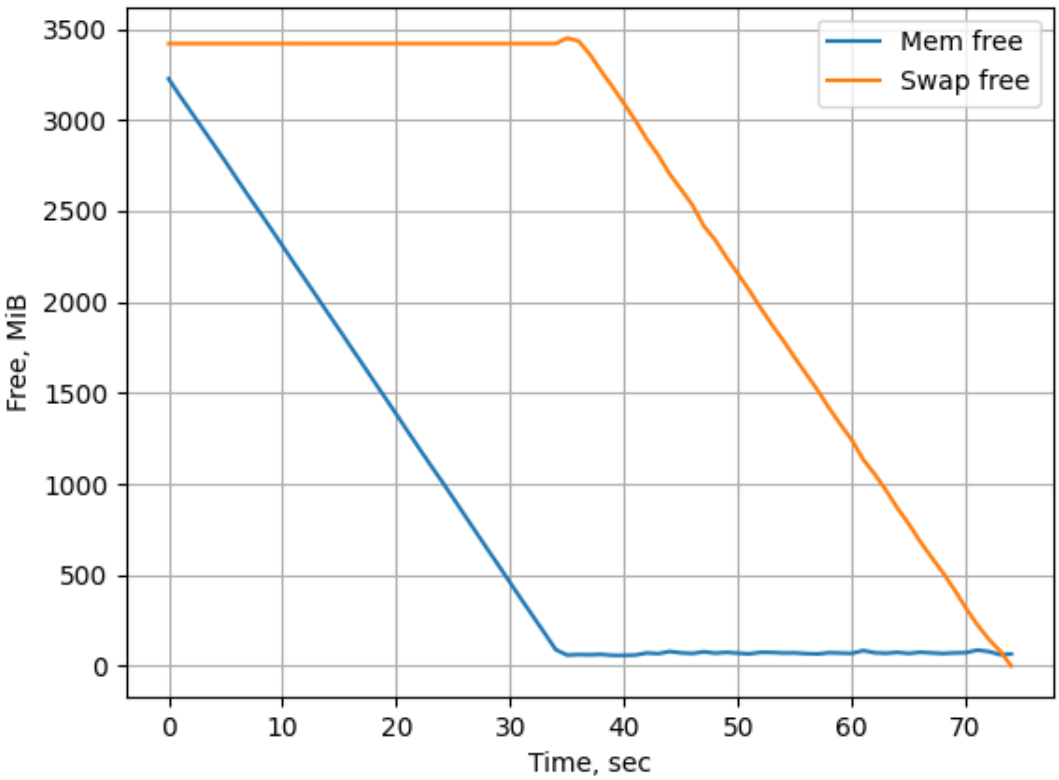

Рис. 1: Свободная оперативная память при запущенном процессе **mem.sh**

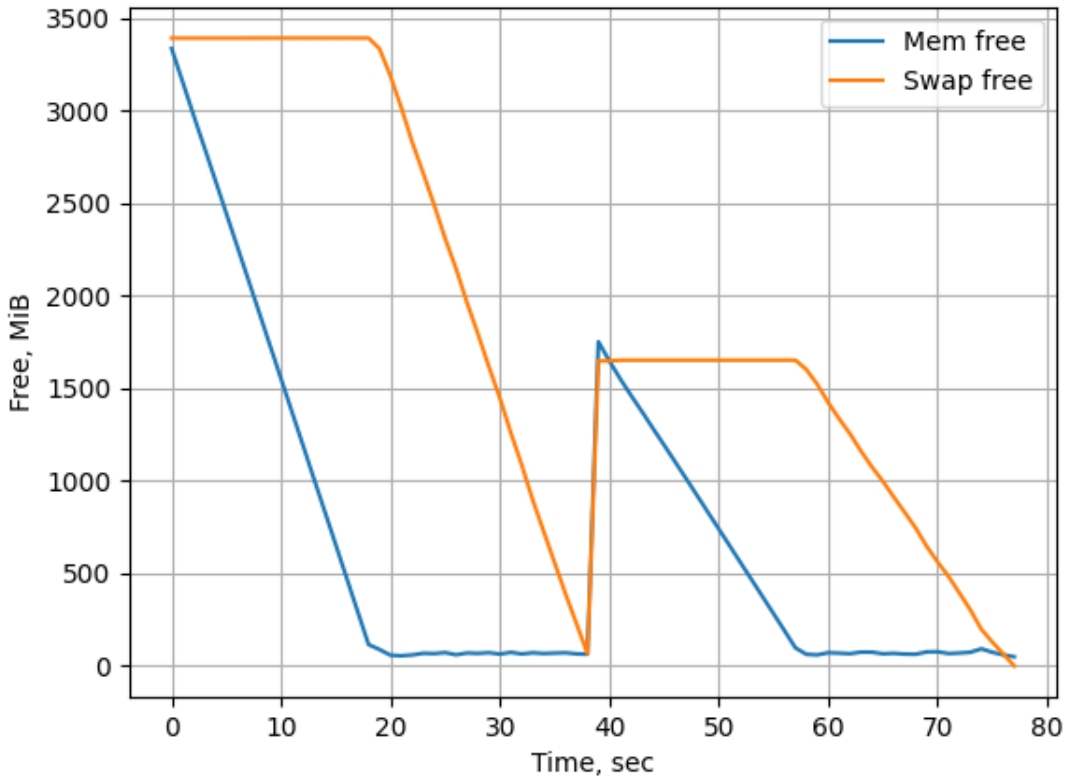

Рис. 2: Свободная оперативная память при запущенных процессах **mem.sh** и **mem2.sh**