Отчёт по лабораторной работе №5

Конфигурация операционной системы:

* Общий объём оперативной памяти 1817,2 MB
* Объем раздела подкачки 820 MB
* Размер страницы виртуальной памяти 4096 B
* Объем свободной физической памяти в ненагруженной системе 1663 MB
* Объем свободного пространства в разделе подкачки в ненагруженной системе 710 MB

Эксперимент 1

1. Запустили **mem.bash**. Процесс аварийно завершился с последним значением размера массива, равным 3\*107. Последние сообщения системного журнала:

[18409.660501] Out of memory: Killed process 4462 (bash) total-vm:2609208kB, anon-rss:1666108kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:4736kB oom\_score\_adj:0

[18409.776845] oom\_reaper: reaped process 4462 (bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Процесс на момент остановки использовал 2609208 KB памяти, 1666108 KB из которых в оперативной памяти. Можно сделать вывод, что один элемента массива занимает приблизительно 80 байт памяти.

Приведём график зависимости параметров %MEM и %CPU от времени работы скрипта.

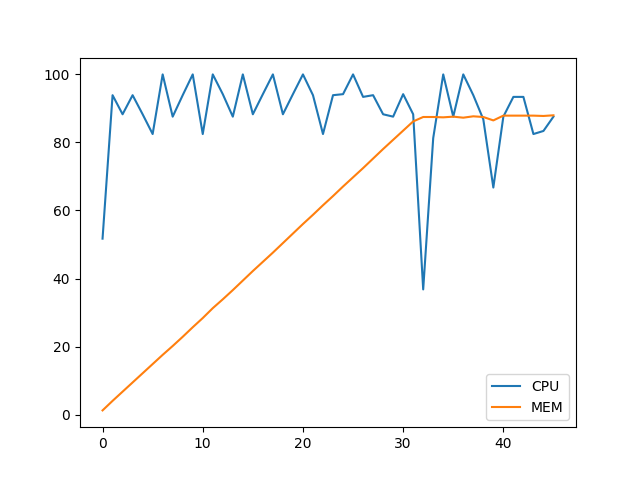
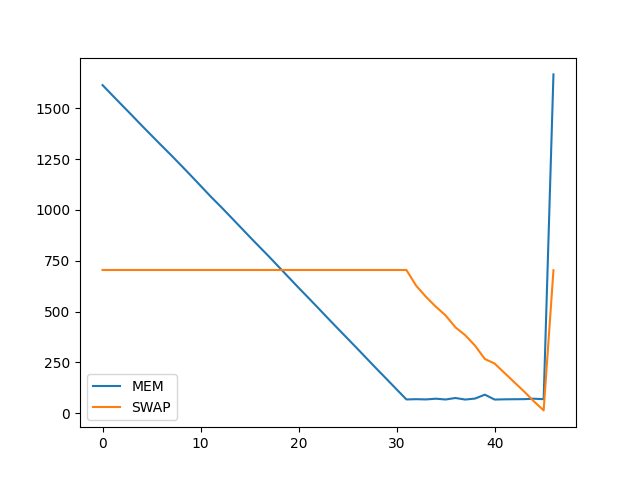


График изменения объёма свободной физической памяти и свободного пространства в разделе подкачки.



Как видно из графиков, объём свободной физической памяти равномерно убывает, а %MEM процесса соответственно растёт. Когда оперативной памяти становится недостаточно, часть страниц из неё переносится в раздел подкачки. Процессор практически всё время выполнял команды, простаивая только при выделении дополнительной памяти. При переносе страниц в раздел подкачки произошло самое большое падение нагрузки.

Верхние 5 процессов: изначально были запущены только системные процессы, **mem.bash** сразу поднялся на первую позицию дальше изменений не было вплоть до остановки процесса.

1. Запустили 2 скрипта – **mem.bash** и **mem2.bash.** Один процесс аварийно завершился с последним значением размера массива, равным 1.5\*107. Второй продолжил работу пока значение размера не достигло 3\*107, после чего произошла ещё одна аварийная остановка. Можно заметить, что процессы потребляли ресурсы в одинаковом темпе, то есть в момент первой аварийной остановки массивы обоих состояли из 1.5\*107 элементов, следовательно вся доступная память закончилась.

Приведём график зависимости параметров %MEM и %CPU от времени работы скриптов. Сообщения в системном журнале:

[19396.898384] Out of memory: Killed process 4780 (bash) total-vm:2608020kB, anon-rss:1665092kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:4740kB oom\_score\_adj:0

[19397.035541] oom\_reaper: reaped process 4780 (bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

и

[19378.069890] Out of memory: Killed process 4781 (bash) total-vm:1418172kB, anon-rss:832984kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:2416kB oom\_score\_adj:0

[19378.142169] oom\_reaper: reaped process 4781 (bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Что соотносится с результатами первого этапа.

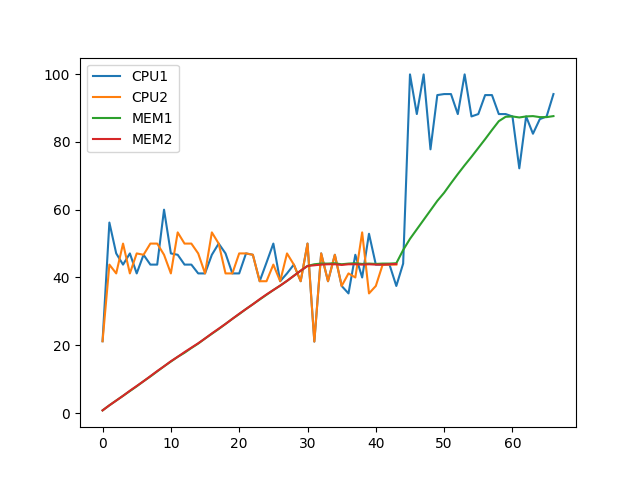
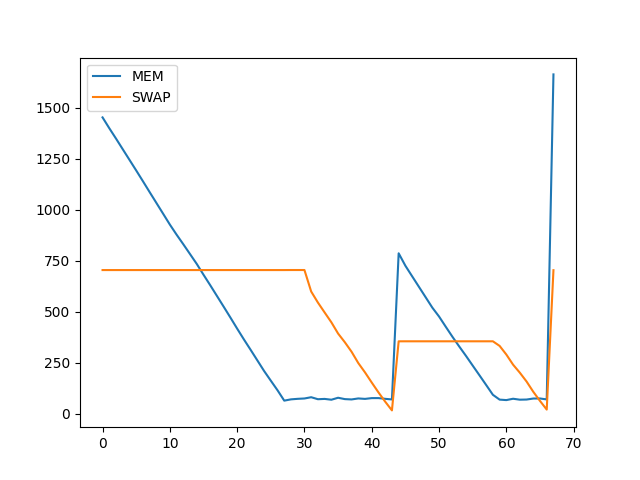


График изменения объёма свободной физической памяти и свободного пространства в разделе подкачки.



Из графиков можно заметить, что сначала оба процесса одинаково потребляли память и нагружали процессор. Затем место в оперативной памяти закончилось, и часть страниц перенеслись в раздел подкачки. Когда память закончилась, один из процессов аварийно завершился. Процессы использовали одинаковое количество памяти, поэтому после остановки одного, физическая память и раздел подкачки наполовину освободились. В оставшееся время работал только один процесс, поэтому график соответствует первому этапу.

Верхние 5 процессов: изначально были запущены только системные процессы, **mem.bash** и **mem2.bash** сразу поднялись на первые 2 позиции и постоянно чередовались, затем остался только **mem.bash** на первой позиции, дальше изменений не было вплоть до остановки процесса.

Вывод:

Было проанализировано изменение параметров, выводимых утилитой **top** при работе одного и нескольких экземпляров скрипта. Наблюдаемое поведение механизмов управления памятью соответствует теоретическим основам.

Эксперимент №2

Из первого эксперимента получаем, что при запуске 1 скрипта максимальное значение N1 равно 3\*107.

K = 10, N = 3\*106 – все процессы как и ожидалось успешно завершились.

K = 30, N = 3\*106 – произошла аварийная остановка, так как процессы требуют больше доступного количества памяти.

K = 30, N = 1.14\*106 – все процессы успешно завершились. Максимально возможное N больше, чем N1/K. Это происходит, потому что все K процессов не могут работать одновременно. Процессы выполняются асинхронно, поэтому некоторые успевают завершить свою работу, следовательно требуют меньше памяти.

Вывод:

В ходе эксперимента были определены и проанализированы граничные значения потребления памяти, обеспечивающие безаварийную работу для регулярных процессов, запускающихся с заданной интенсивностью.