Отчёт по лабораторной работе №5: «Управление памятью в ОС Linux»

Выполнил: Трещёв Артём Сергеевич, M32341

Примечание: здесь и далее используются обозначения

Gb – Гигабайты

Mb – Мегабайты

b - байты

1. **Данные о текущей конфигурации операционной системы**

Общий объём оперативной памяти: 4.8Gb (5049Mb, из них под саму память отведено 3.8Gb(3975Mb), остальное под раздел подкачки файлов)

Объём раздела подкачки: 1.0Gb (1073Mb)

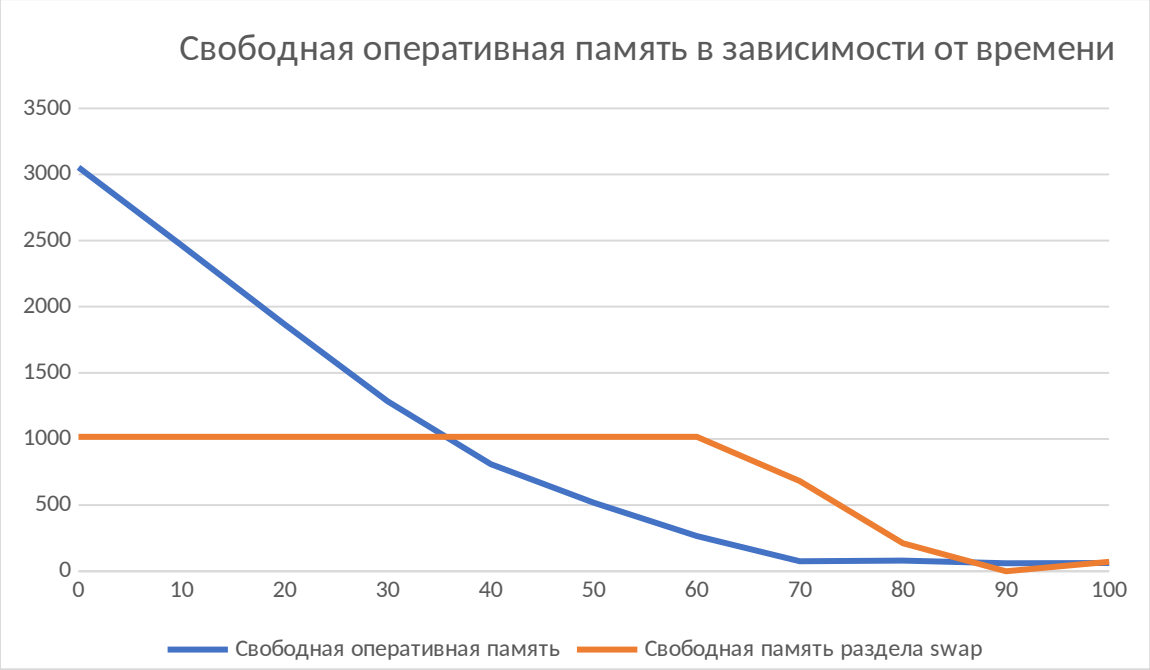
Размер страницы виртуальной памяти: 4096b

Объём свободной физической памяти в ненагруженной системе: 3.7Gb(3842Mb)

Объём свободного пространства в разделе подкачки в ненагруженной системе: 1Gb(1073Mb)

2. **Эксперимент №1**

Этап 1:



Здесь по горизонтальной шкале время в секундах, вертикальная – размер в Мегабайтах

Как видно из графиков, процессор изначально не занимал много места в оперативной памяти, однако со временем его работы, память, которую он требовал от операционной системы, нарастала. Когда оперативная память закончилась, операционная система начала давать процессу память из раздела подкачки, которая до этого была почти полностью свободной. Когда вся свободная память закончилась, процесс был прерван, и память, занятая им, начала освобождаться (на графиках это момент времени 100 секунд). На протяжении всего времени процесс mem.sh занимал верхнюю строчку в таблице top, так как этот процесс требовал больше всего процессорного времени, остальные же процессы по сравнению с данным требовали меньше 0.1% процессорного времени, поэтому находились ниже в таблице, однако выше всех из них находился процесс top, следивший за процессами, ниже них – системные процессы, в том числе bash

Последняя запись в report.log: 60000000

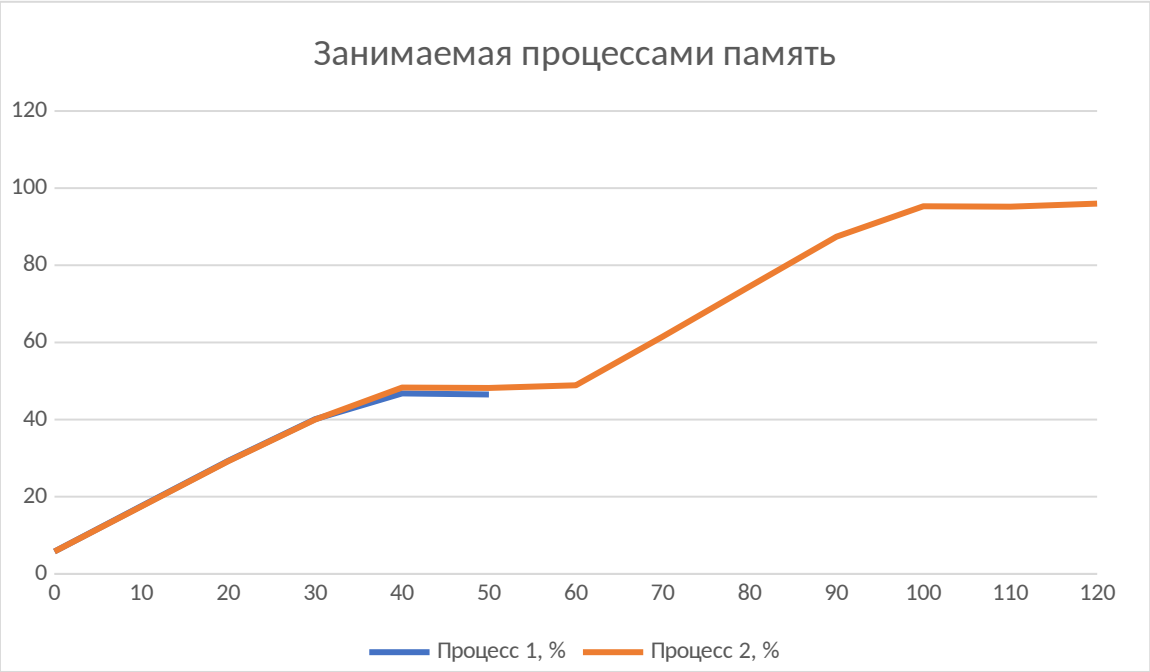
Записи о скрипте в системном журнале:

oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=bash,pid=11064,uid=1000

out of memory: Killed process 11064 (bash) total-vm:4747176kB, anon-rss:3694468kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:9332kB oom_score_adj:0

oom_reaper: reaped process 11064 (bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Этап 2:



Здесь, как и на предыдущем этапе по горизонтали отложено время в секундах с момента запуска скрипта, по вертикали на первом графике объём памяти в Мб, во втором – процент занимаемой оперативной памяти. Как видно, изначально процессы шли друг с другом и «кушали» одинаковое количество памяти, потом, так как свободной памяти не осталось, был прерван процесс 1, появилось свободное место, которое уже начал занимать процесс 2. Эти 2 процесса занимали всё время верхнюю строчку в top (иногда меняясь местами в зависимости от того процессорного времени, которое они имели). После того как процесс 2 забрал всю освободившуюся память, он был прерван

Последняя запись в report.log : 30000000

Последняя запись в report2.log: 61000000

Записи о первом скрипте в системном журнале:

oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=bash,pid=11125,uid=1000

Out of memory: Killed process 11125 (bash) total-vm:2397980kB, anon-rss:1842612kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:4724kB oom_score_adj:0

oom_reaper: reaped process 11125 (bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Записи о втором скрипте в системном журнале:

oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/,task=bash,pid=11126,uid=1000

Out of memory: Killed process 11126 (bash) total-vm:4778444kB, anon-rss:3725876kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:9384kB oom_score_adj:0

oom_reaper: reaped process 11126 (bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Выводы:

Как мы видим, при активном требовании процесса большого количества памяти, изначально память даётся оперативная, а когда она заканчивается, за дело берётся раздел подкачки. Если процесс забирает всю возможную память, то специальный процесс oom-killer, определит, что какой-то процесс забрал всю память, а так как свободной нет, то пошлёт сигнал sigterm процессу

3. **Эксперимент №2**

Как мы знаем из предыдущего эксперимента, прежде чем процесс mem.sh аварийно завершит свою работу, длина массива достигнет 60 000 000. Сначала мы запускаем 10 процессов каждую секунду, но они работают безаварийно, если поставить ограничение на 6 000 000 элементов в массиве. Если же запустить 30 процессов, то суммарно у нас будет в памяти не 6 000 000 * 10 элементов, а 6 000 000 * 30 элементов, что в 3 раза больше 60 000 000, которое является максимальным значением, при котором не происходит аварийных завершений работы.

Поэтому следуя логики стоит ограничить максимальную длину массива не на 6 000 000, а на 2 000 000 элементов, чтобы 30 процессов отработали без аварийной остановки. Однако, при таких параметрах запусках, я увидел, что остаётся ещё много свободной оперативной памяти, а раздел подкачки почти не затрагивается, поэтому я изменил максимальный размер на 3 000 000 элементов, и получил всё ещё без аварийную работу 30 процессов, но при максимальной нагрузке свободной памяти в разделе подкачки оставалось примерно 20 Мбайт, что много меньше общего объёма swap, поэтому 3 000 000 элементов считаю предельно допустимым.