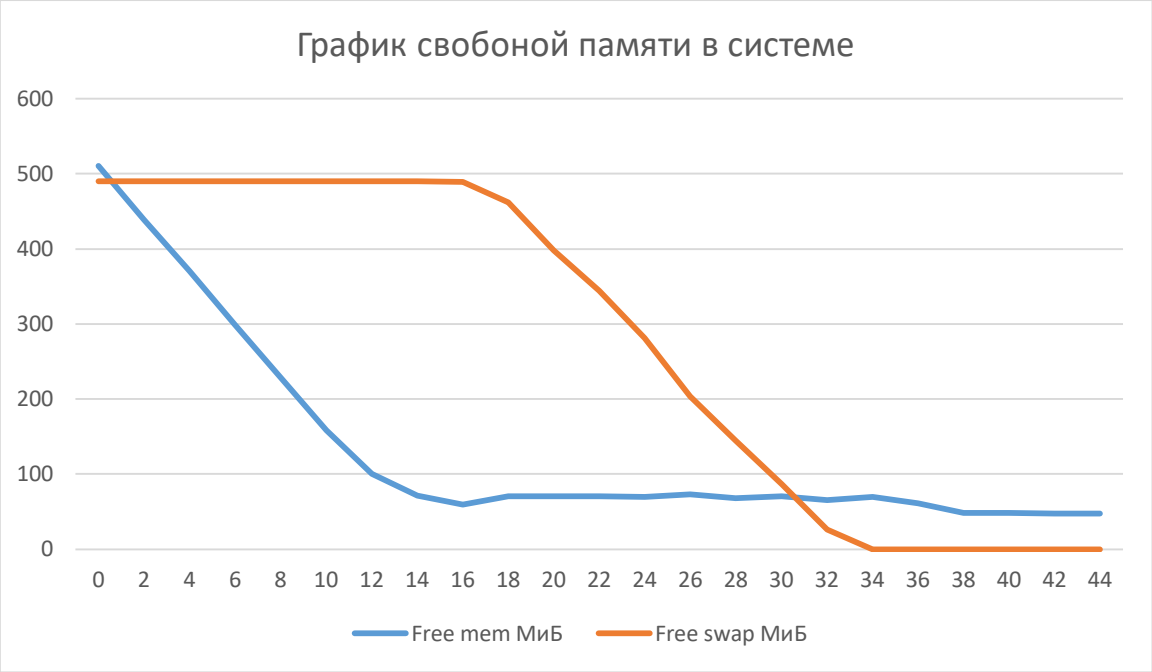
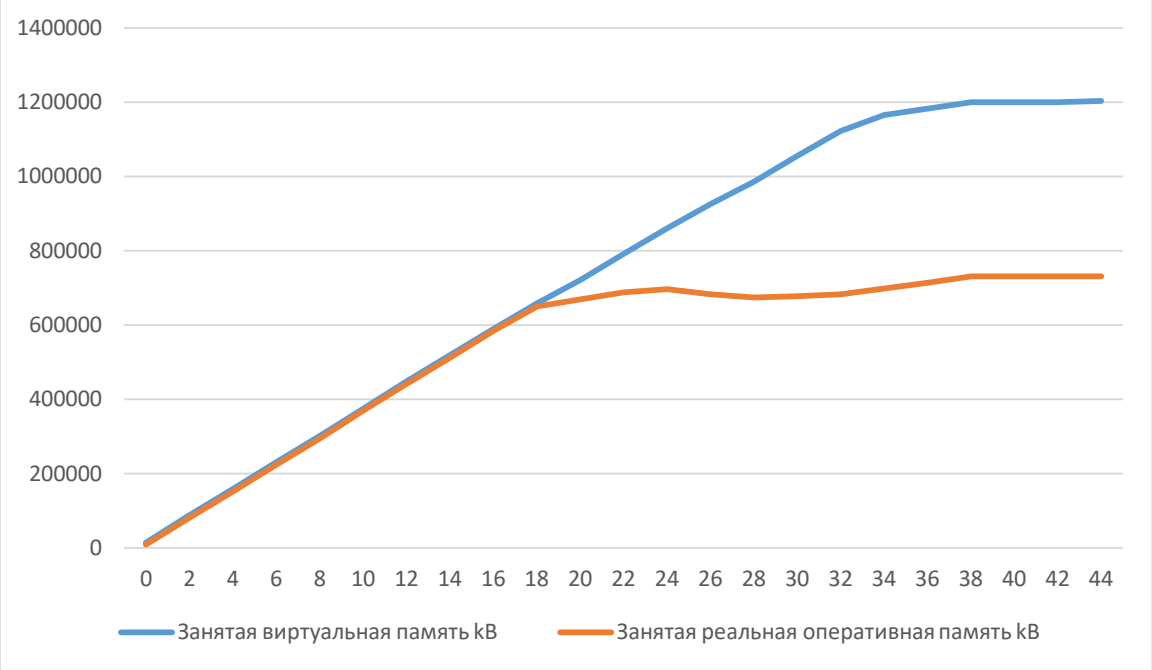
Отчет к лабораторной работе

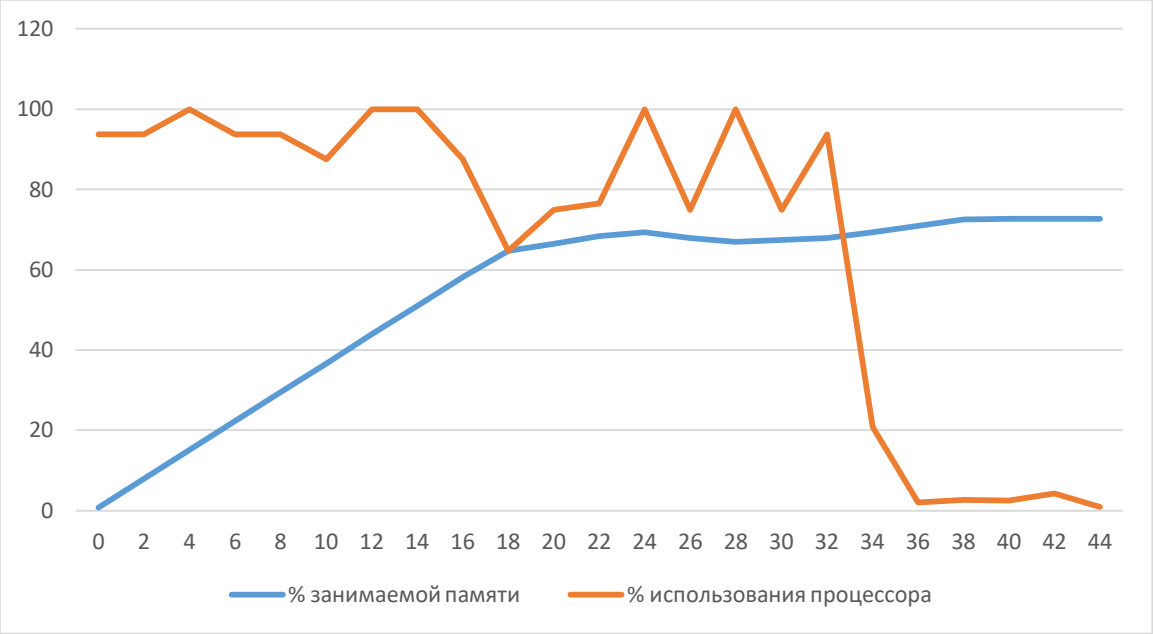
К отчету прикреплен код всех скриптов и файлы с полученными данными (был написан скрипт просматривающий состоянии системы, который запускался в фоне при запуске экспериментальных скриптов).

Параметры системы:

1) getconf PAGE_SIZE 4096
2) Общий объем оперативной памяти: 1004848 kB
3) Объем раздела подкачки: 945416 kB
4) Размер страницы виртуальной памяти: 4096 kB
5) Объем свободной физической памяти в ненагруженной системе: 342344 kB
6) Объем свободного пространства в разделе подкачки в ненагруженной системе: 727928 kB
7) Суммарно свободной памяти в ненагруженном состоянии: 1287760 kB







Секунда	COMMAND
0	mem.bash system kthreadd rcu_gp rcu_par+
6	mem.bash system kthreadd rcu_gp rcu_par+
12	mem.bash system kthreadd rcu_gp rcu_par+
18	mem.bash kswapd0 system kthreadd rcu_gp
24	mem.bash kswapd0 kworker+ top systemd
30	mem.bash kswapd0 system kthreadd rcu_gp
36	kswapd0 mem.bash Xorg gnome-t+ gnome-s+
44	kswapd0 snapd

	mem.bash loop9 gnome-s+

Последние записи в логе:

[20515.131205] Out of memory: Killed process 26493 (mem.bash) total-vm:1203572kB, anon-rss:725204kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:2392kB oom_score_adj:0

[20515.421228] oom_reaper: reaped process 26493 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Процессу было выделено 1203572kB виртуальной памяти, что практически равно суммарно свободной памяти в ненагруженном состоянии.

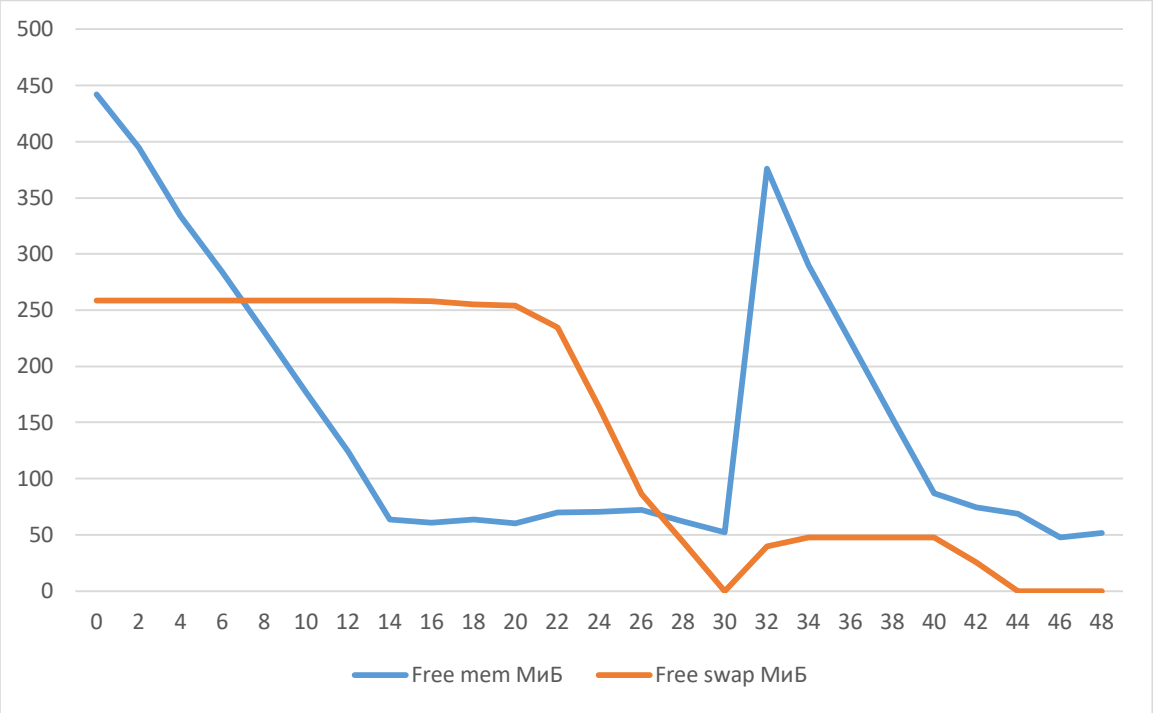
Последняя запись в report.log:

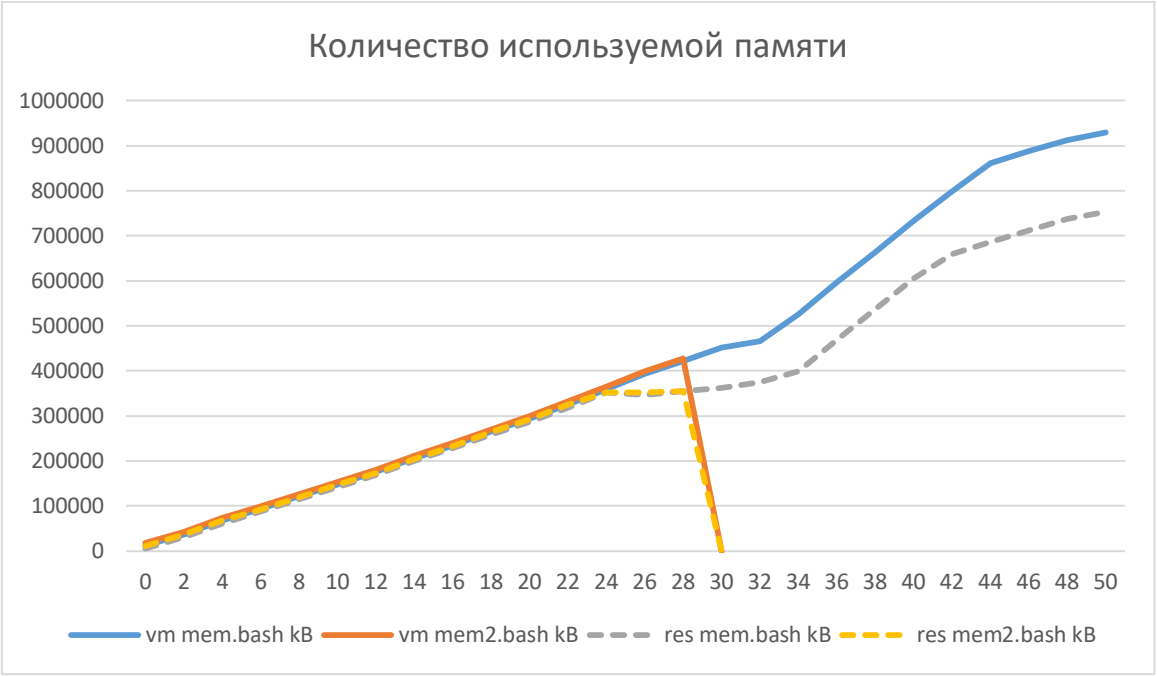
15000010

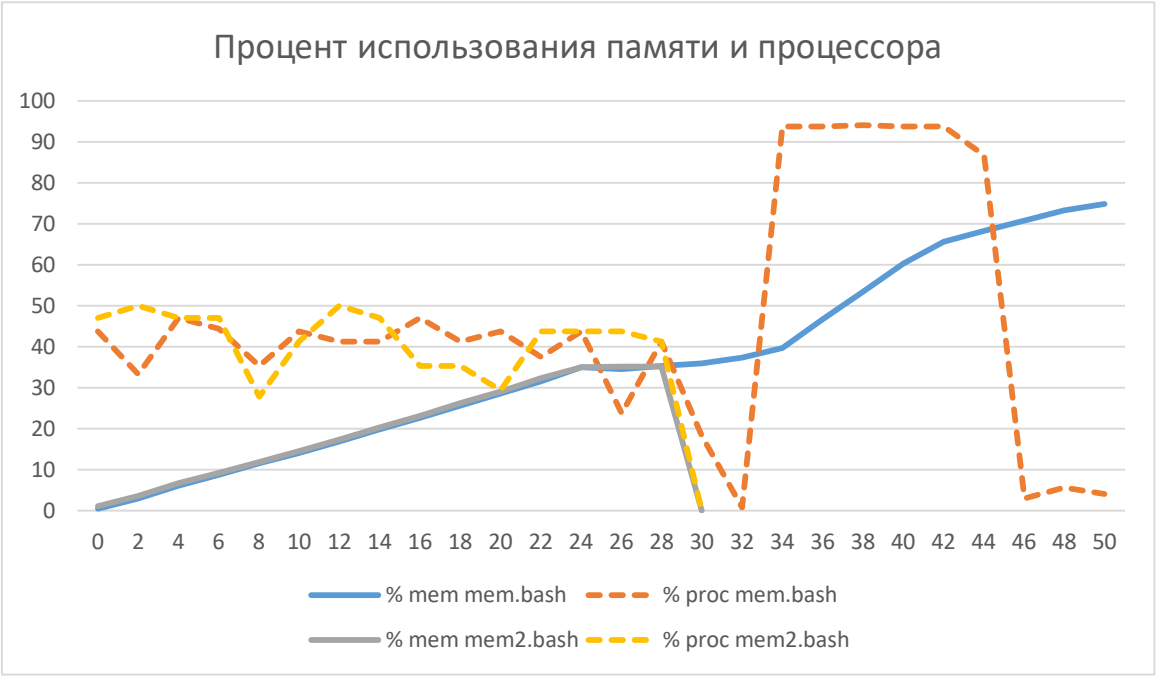
Вывод: из графиков видим, что процесс поглощал память настоящей оперативной памяти, когда она закончилась процесс начал занимать память в разделе подкачки, нагружая систему механизмом swap, когда и эта память закончилась, процесс пытался найти еще памяти, но т.к. её не нашлось, процесс был аварийно завершён.

Второй этап:

Запуск двух копий mem.bash одновременно.







Секунда	COMMAND
0	mem2.bash mem.bash top system kthreadd
6	mem2.ba mem.bash gnome-s+ top systemd
12	mem.bash mem2.ba systemd kthreadd rcu_gp
18	mem.bash

	mem2.ba+ top systemd kthreadd
24	mem.bash mem2.ba kswapd0 top systemd
30	kswapd0 loop4 snap-st+ loop8 rcu_sch+
36	mem.bash systemd kthreadd rcu_gp rcu_par+
42	mem.bash systemd kthreadd rcu_gp rcu_par+
46	mem.bash systemd kthreadd rcu_gp rcu_par+
50	kswapd0 snap-st+ loop4 top mem.bash

Последние записи в логе:

[12842.289893] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000.service,task=mem2.bash,pid=8321,uid=1000

[12842.289900] Out of memory: Killed process 8321 (mem2.bash) total-vm:469388kB, anon-rss:378148kB, file-rss:0kB, shmem-rss:0kB, UID:1000 pgtables:956kB oom_score_adj:0

[12842.432210] oom_reaper: reaped process 8321 (mem2.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

[12988.934037] [8320] 1000 8320 232319 187951 1896448 42043 0 mem.bash

[12988.934041] oom-kill:constraint=CONSTRAINT_NONE,nodemask=(null),cpuset=/,mems_allowed=0,global_oom,task_memcg=/user.slice/user-1000.slice/user@1000.service,task=mem.bash,pid=8320,uid=1000

[12988.934066] Out of memory: Killed process 8320 (mem.bash) total-vm:929276kB, anon-rss:751796kB, file-rss:8kB, shmem-rss:0kB, UID:1000 pgtables:1852kB oom_score_adj:0

[12989.184387] oom_reaper: reaped process 8320 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Последняя запись в report.log: 5000010

Последняя запись в report2.log: 11000010

Вывод:

Из полученных данных можно сделать вывод о том как отработали эти процессы. В начале им обоим одинаково выделялась оперативная память, когда она закончилась выделялась память из раздела подкачки, затем, когда и она закончилась, второй процесс аварийно завершился, т.к. он сделал запрос на выделение ему памяти, а памяти не было, первый продолжил работать штатно, т.к. теперь освободилась память выделенная второму процессу. Он также поглотил всю оперативную память, а затем и память из раздела подкачки и аварийно завершился аналогично эксперименту на первом этапе.

Эксперимент 2:

Возьмем за максимум элементов 15000000 по первому эксперементу.

1) N = 1500000 K=10 все процессы были завершены корректно. Это объясняется тем, что процессы могу суммарно достигнуть критических 15000000 элементов, если будут выполняться одновременно и максимум наберут в один момент, и то этого не достаточно, т.к. суммарно они должны преодолеть эту отметку (из первого эксперимента выяснилось, что даже для одного процесса 15000000 элементов достижимо).
2) N = 1500000 K=30 14 процессов завершили работу аварийно. Это объясняется тем, что при 30 процессах суммарное количество элементов в одно время может превышать критическое значение, при котором заполняется вся память, и в такие моменты приходится авайрино завершать некоторые процессы.
3) Будем находить максимальное N, при котором все процессы завершаются корректно.
   То есть такое наибольшее N при котором процессы успевают завершаться и освобождать память так, чтобы вся память не заполнялась.
   N=1000000 K=30 все завершилось корректно.
   N=1200000 K=30 12 процессов завершились аварийно.
   N=1100000 K= 30 все процессы завершены корректно.
   N=1150000 K= 30 все процессы завершены корректно.
   Такое N находится в промежутке от 1150000 до 1200000. Более точный поиск бессмысленный, т.к. значение может меняться от незначительных изменений в состоянии системы.

Вывод:

Были изучены механизмы управления памятью в Linux, поведение процессов в критических ситуациях, а также механизмы слежения за потребляемыми ресурсами.