Лабораторная работа #5
Управление памятью в ОС Linux

Ключжев Игорь
Группа №M3203
Преподаватель: Титова А.В.

6 декабря 2020 г.

1 Цель работы

1. Использование утилиты top для мониторинга параметров памяти

2. Использование имитационных экспериментов для анализа работы механизмов управления памятью.

2 Основные источники данных о состоянии памяти вычислительного узла

- команда **free**
- файл /**proc/meminfo**
- файл /**proc/[PID]/statm**

- утилита **top**

3 Задание на лабораторную работу

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.

4 Текущая конфигурация операционной системы

- Общий объём оперативной памяти - **486196 kB**
- Объём раздела подкачки - **839676 kB**
- Размер страницы виртуальной памяти - **2048 kB**
- Объём свободной физической памяти в ненагруженной системе - **124844 kB**
- Объём свободного пространства в разделе подкачки в ненагруженной системе - - **839676 kB**

5 Численные результаты

Эксперимент №1

Первый этап

Последняя запись в системном журнале:

[1296.113259] **Out of memory: Killed process** 1445 **(mem.bash) total-vm:**1338152 **kB, anon-res:**340724 **kB, file-rss:**0 **kB, shmem-rss:**0 **kB, UID:**0

[1296.214684] **oom_reaper: reaped process** 1445 **(mem.bash), now anon-res:**0 **kB, file-rss:**0 **kB, shmem-rss:**0 **kB**

Последняя запись в файле **report.log** - 14000000.

	MiB Mem				MiB Swap			
Time, s	Total	Free	Used	buff/cache	Total	Free	Used	avail Mem
0	474,8	75,1	166,7	233,0	820,0	820,0	0,0	292,3
1	474,8	58,3	183,5	233,0	820,0	820,0	0,0	275,5
2	474,8	7,9	257,6	209,4	820,0	818,2	1,8	203,0
3	474,8	7,8	295,1	171,9	820,0	816,7	3,3	166,8
4	474,8	7,7	329,3	137,8	820,0	816,5	3,5	132,7
5	474,8	8,3	364,0	102,5	820,0	816,0	4,0	98,1
6	474,8	7,2	405,2	62,3	820,0	815,0	5,0	56,9
7	474,8	7,5	420,4	46,9	820,0	782,0	38,0	41,7
8	474,8	7,4	423,1	44,4	820,0	746,0	74,0	39,1
9	474,8	8,8	423,3	42,7	820,0	708,5	111,5	38,8
10	474,8	8,4	424,3	42,1	820,0	672,5	147,5	37,9
11	474,8	9,0	412,4	53,4	820,0	623,2	196,8	49,8
12	474,8	7,6	414,7	52,4	820,0	589,0	231,0	46,8
13	474,8	6,8	414,3	53,7	820,0	556,2	263,8	47,3
14	474,8	8,5	417,3	49,0	820,0	521,3	298,7	44,3
15	474,8	8,2	420,1	46,5	820,0	490,7	329,3	41,4
16	474,8	7,8	421,5	45,5	820,0	452,5	367,5	40,1
17	474,8	5,4	425,5	43,9	820,0	421,7	398,3	36,2
18	474,8	7,5	425,7	41,6	820,0	385,7	434,3	36,0
19	474,8	7,6	424,6	42,6	820,0	348,7	471,3	37,1
20	474,8	6,8	424,7	43,4	820,0	310,0	510,0	37,2
21	474,8	7,8	426,2	40,8	820,0	276,1	543,9	35,6
22	474,8	7,0	425,8	42,1	820,0	235,2	584,8	36,1
23	474,8	8,0	425,4	41,4	820,0	198,7	621,3	36,6
24	474,8	8,3	409,5	57,0	820,0	144,0	676,0	52,5
25	474,8	4,9	410,8	59,1	820,0	115,5	704,5	51,1
26	474,8	5,1	417,8	51,9	820,0	86,5	733,5	44,1
27	474,8	4,9	421,7	48,2	820,0	54,7	765,3	40,3
28	474,8	7,0	422,2	45,6	820,0	17,6	802,4	39,7

Второй этап

Последнии записи в системном журнале:

[2742.068923] [1519] 0 1519 193727 43404 1187840 94754 0 mem.bash

[2742.069156] [1520] 0 1520 196202 45304 1200128 95337 0 mem2.bash

[2742.069659] Out of memory: Killed process 1520 (mem2.bash) total-vm:784808kB, anon-rss:181216kB, file-rss:0kB, shmem-rss:0kB, UID:0

[2742.100445] oom_reaper: reaped process 1520 (mem2.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

[2804.808992] [1519] 0 1519 335165 89483 2322432 190116 0 mem.bash

[2804.809297] Out of memory: Killed process 1519 (mem.bash) total-vm:1340660kB, anon-rss:357928kB, file-rss:4kB, shmem-rss:0kB, UID:0

[2804.889676] oom_reaper: reaped process 1519 (mem.bash), now anon-rss:0kB, file-rss:0kB, shmem-rss:0kB

Последняя запись в файле **report.log** - 14000000.

Последняя запись в файле **report2.log** - 7000000.

	MiB Mem				MiB Swap			
Time, s	Total	Free	Used	buff/cache	Total	Free	Used	avail Mem
0	474,8	249,8	126,6	98,4	820,0	741,9	78,1	334,7
1	474,8	207,7	168,7	98,4	820,0	741,9	78,1	292,7
2	474,8	166,4	210,0	98,4	820,0	741,9	78,1	251,4
3	474,8	126,2	250,2	98,4	820,0	741,9	78,1	211,2
4	474,8	114,0	262,3	98,5	820,0	741,9	78,1	199,0
5	474,8	90,2	286,2	98,4	820,0	741,9	78,1	175,1
6	474,8	51,6	324,8	98,4	820,0	741,9	78,1	136,5
7	474,8	11,7	364,7	98,4	820,0	741,9	78,1	96,7
8	474,8	5,3	400,9	68,6	820,0	735,1	84,9	60,4
9	474,8	5,0	415,2	54,6	820,0	706,3	113,7	46,1
10	474,8	4,9	421,2	48,7	820,0	672,7	147,3	40,2
11	474,8	5,7	424,6	44,5	820,0	632,5	187,5	36,7
12	474,8	5,0	427,2	42,6	820,0	595,2	224,8	34,2
13	474,8	5,4	426,6	42,8	820,0	555,7	264,3	34,7
14	474,8	5,8	428,9	40,1	820,0	516,1	303,9	32,4
15	474,8	5,3	427,3	42,2	820,0	479,5	340,5	34,0
16	474,8	5,4	427,3	42,1	820,0	441,5	378,5	34,0
17	474,8	6,0	427,0	41,8	820,0	405,5	414,5	34,4
18	474,8	4,4	422,3	48,1	820,0	361,2	458,8	39,0
19	474,8	5,9	413,7	55,2	820,0	340,7	479,3	47,6
20	474,8	5,7	417,1	52,0	820,0	317,2	502,8	44,2
21	474,8	5,7	415,3	53,8	820,0	271,8	548,2	46,0
22	474,8	5,6	415,2	54,0	820,0	234,7	585,3	46,1
23	474,8	4,9	422,4	47,5	820,0	204,1	615,9	39,0
24	474,8	5,8	424,0	45,0	820,0	165,7	654,3	37,3
25	474,8	5,1	425,3	44,3	820,0	127,9	692,1	36,0
26	474,8	5,9	425,7	43,2	820,0	89,5	730,5	35,7
27	474,8	5,6	428,0	41,2	820,0	53,0	767,0	33,4
28	474,8	5,3	427,1	42,4	820,0	10,9	809,1	34,3

	MiB Mem				MiB Swap			
Time, s	Total	Free	Used	buff/cache	Total	Free	Used	avail Mem
29	474,8	122,5	297,4	55,0	820,0	378,6	441,4	164,0
30	474,8	83,1	336,7	55,0	820,0	378,6	441,4	124,6
31	474,8	44,1	375,6	55,0	820,0	378,6	441,4	85,7
32	474,8	5,1	416,4	53,2	820,0	378,6	441,4	44,8
33	474,8	5,6	409,7	59,5	820,0	351,3	468,7	51,6
34	474,8	5,5	413,6	55,7	820,0	337,9	482,1	47,7
35	474,8	5,3	411,4	58,1	820,0	300,9	519,1	49,9
36	474,8	5,1	417,4	52,3	820,0	268,2	551,8	43,9
37	474,8	5,3	418,9	50,6	820,0	228,0	592,0	42,4
38	474,8	5,9	420,7	48,2	820,0	192,1	627,9	40,6
39	474,8	5,7	423,8	45,4	820,0	154,7	665,3	37,5
40	474,8	5,5	425,3	44,0	820,0	116,9	703,1	36,1
41	474,8	5,7	425,6	43,5	820,0	78,8	741,2	35,7
42	474,8	6,5	425,8	42,5	820,0	39,2	780,8	35,5
43	474,8	5,5	426,8	42,5	820,0	2,6	817,4	34,5
44	474,8	324,8	95,6	54,4	820,0	740,7	79,3	365,7

Эксперимент №2

Во втором эксперименте, при $k = 10$ все процессы завершились в штатном режиме. Но про $k = 30$ несколько процессов завершились аварийно, а несколько удачно. Это происходит потому, что процессы, которые запускаются раньше, занимают всё больше и больше памяти и при этом каждую секунду добавляется ещё один скрипт, который тоже начинает наращивать свой размер и занимать память. В результате этого, первым процессам просто не хватает памяти, чтобы дойти до значения n и завершиться; они не успевают дойти, так как память занимают другие процессы этого же скрипта.

С помощью метода проб и ошибок было подобрано максимальное значение переменной n(размер массива, при котором скрипт закан-

чивает работу), при котором все 30 скриптов завершают свою работу в штатном режиме. $n = 700220$

Таким образом, если $n = 700230$ хотя бы один скрипт заканчивает работу аварийно.

6 Скриншоты тестов для эксперимента 2

```
#!/bin/bash
echo "" > exitLog
rm -f exitLog
touch exitLog
echo "" > startLog
rm -f startLog
touch startLog
k=10
n="$(cat report.log | tail -n 1)"
n=$((n/10))
# n=700210 # max with k=30
for ((i=0; i < k; i++))
do
echo "starting $((i+1))" >> startLog
./newmem.bash $n &
sleep 1
done
```

```
[root@localhost lab5]# bash newMemBashStarter
[root@localhost lab5]# _
```

Рис. 1: **Script n=1400000 k=10**

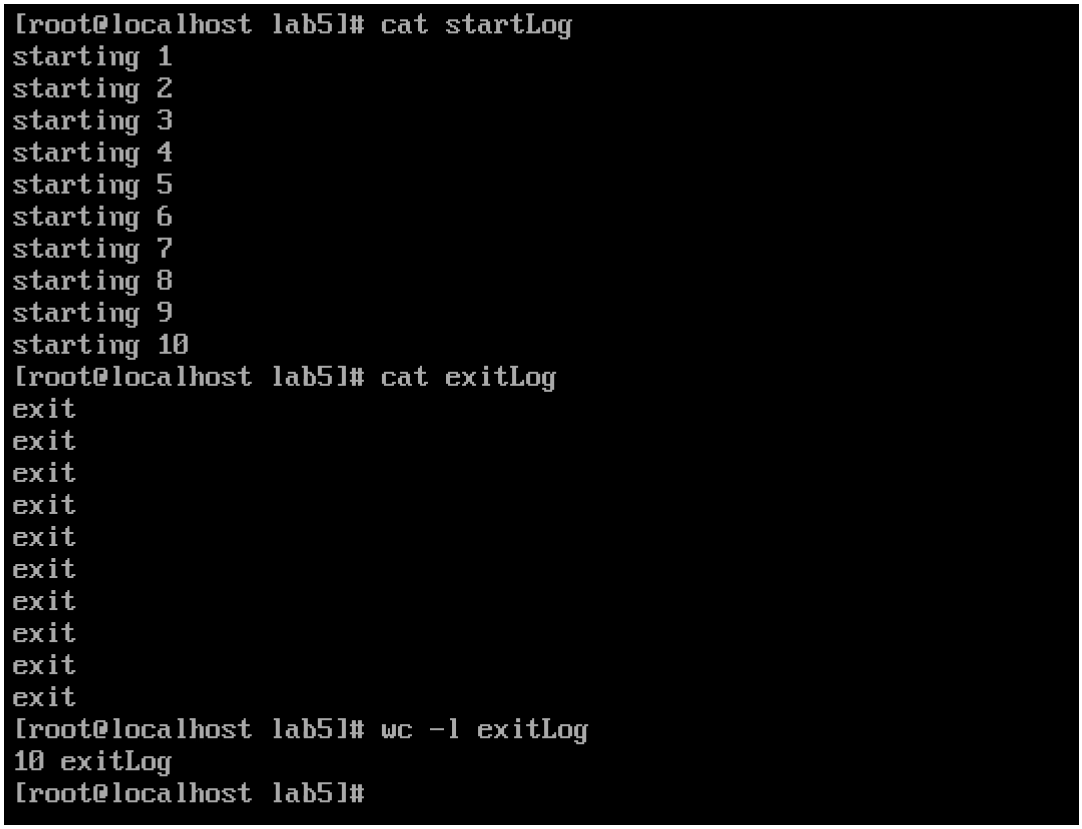


Рис. 2: **StartLog + exitLog n=1400000 k=10**

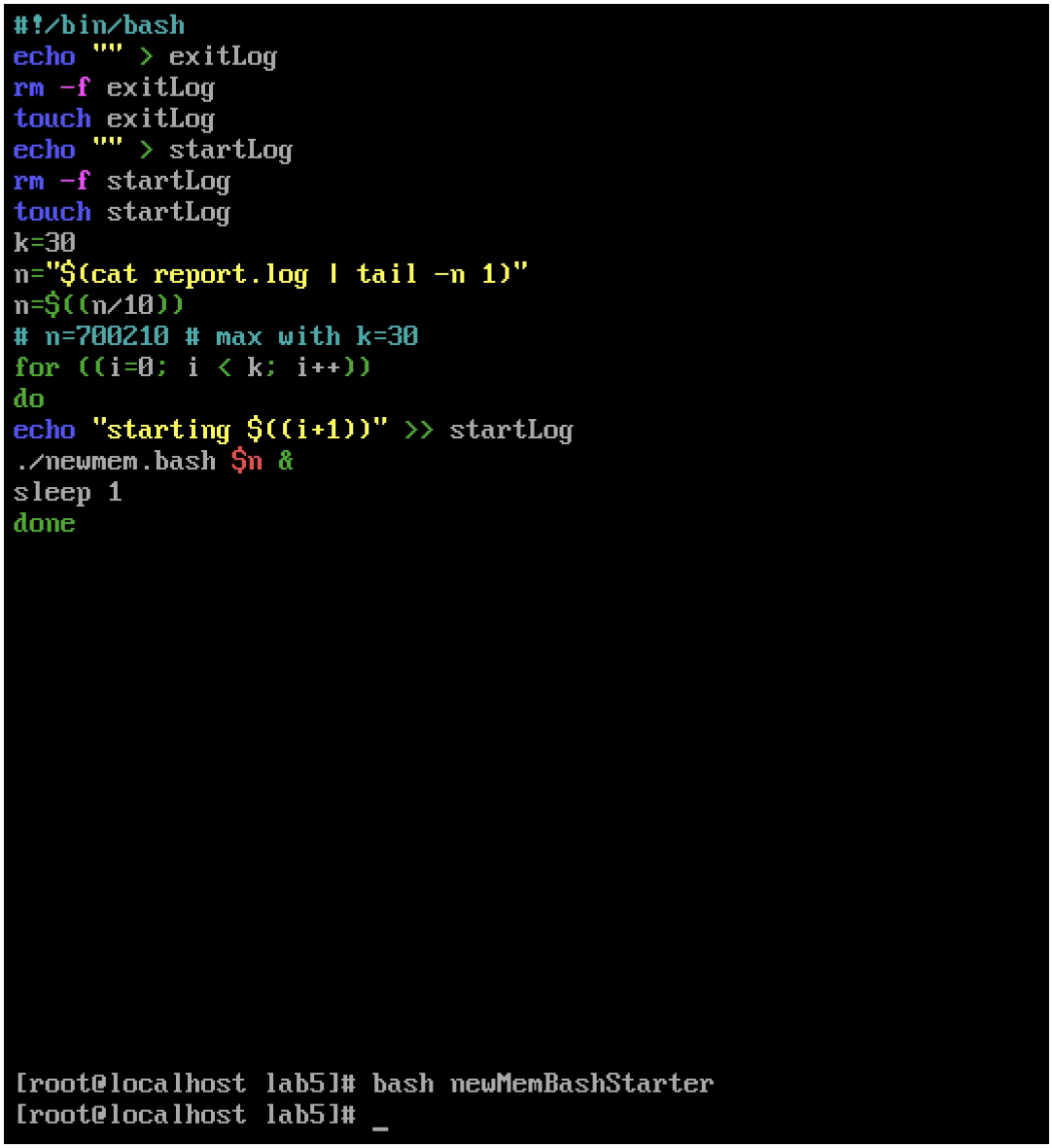


Рис. 3: **Script n=1400000 k=30**

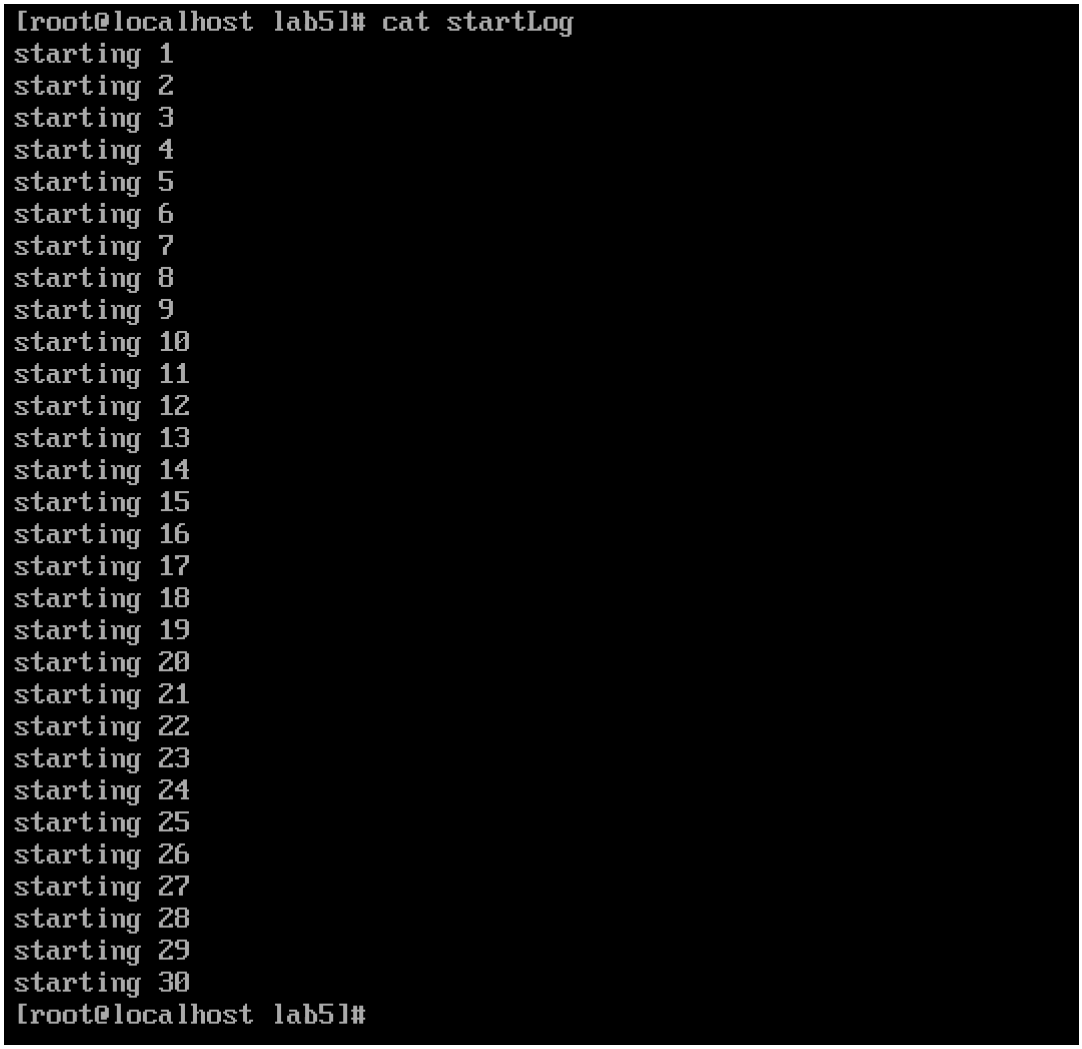


Рис. 4: **StartLog n=1400000 k=30**

Рис. 5: **ExitLog n=1400000 k=30**

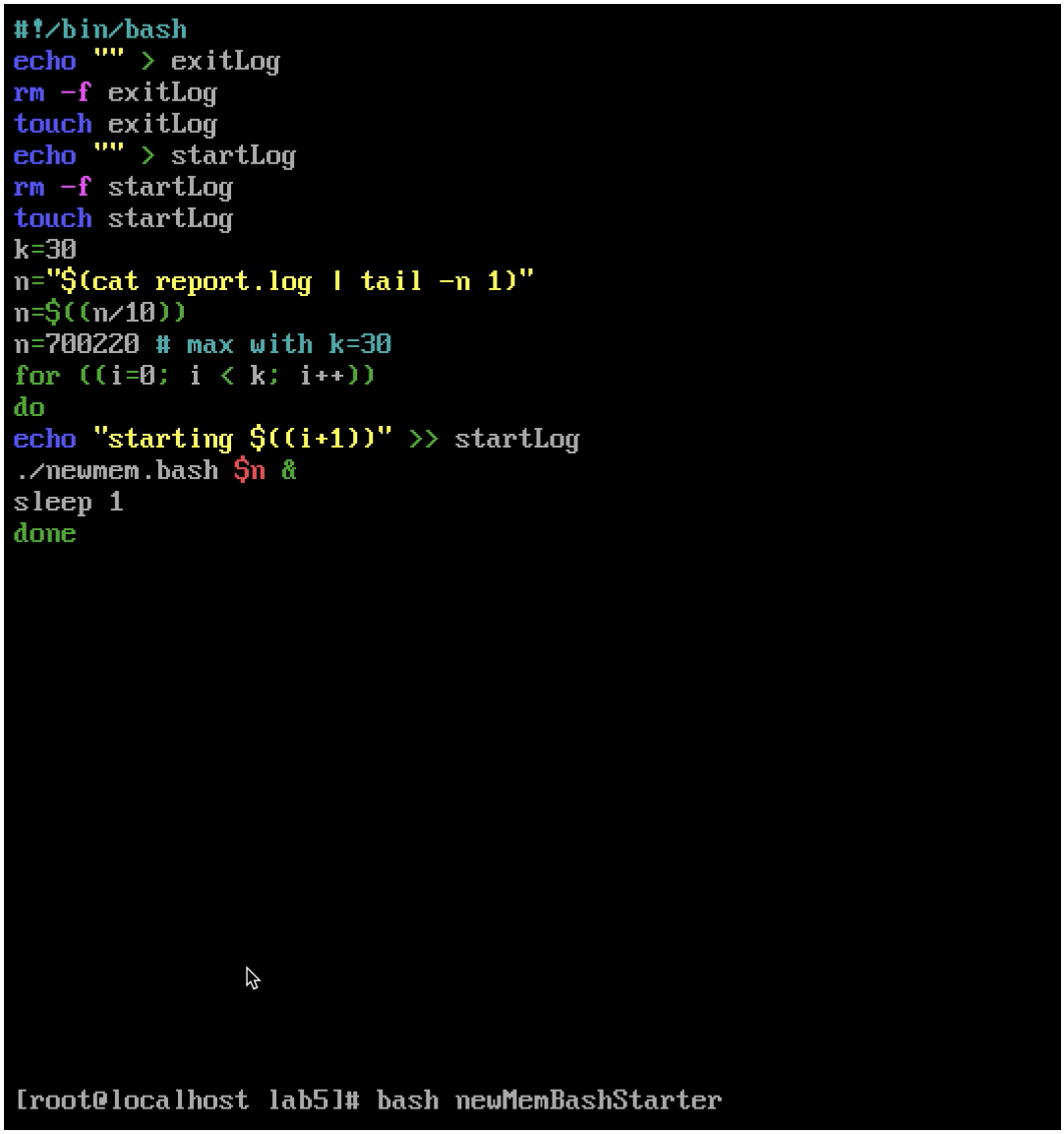


Рис. 6: **Script n=700220 k=30**

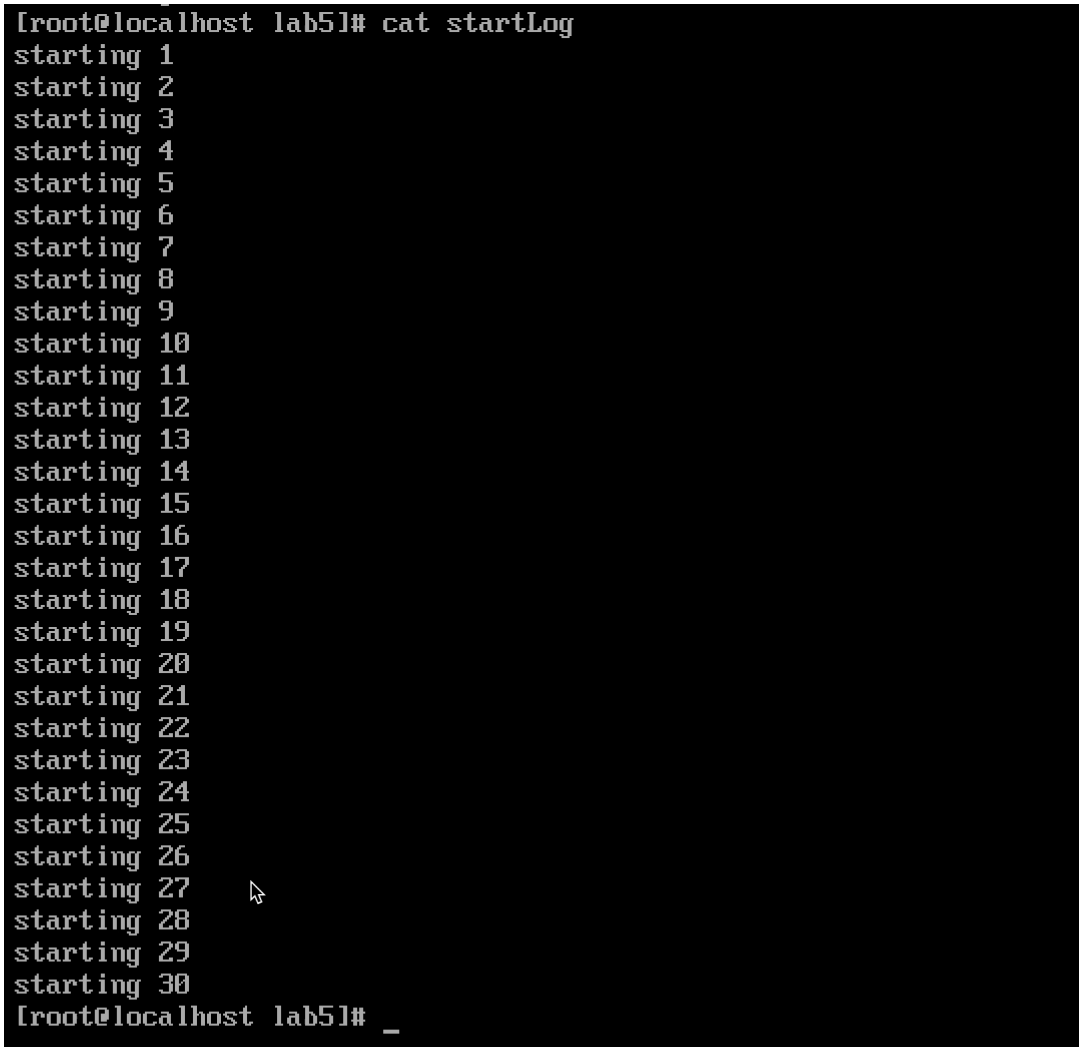


Рис. 7: **StartLog n=700220 k=30**

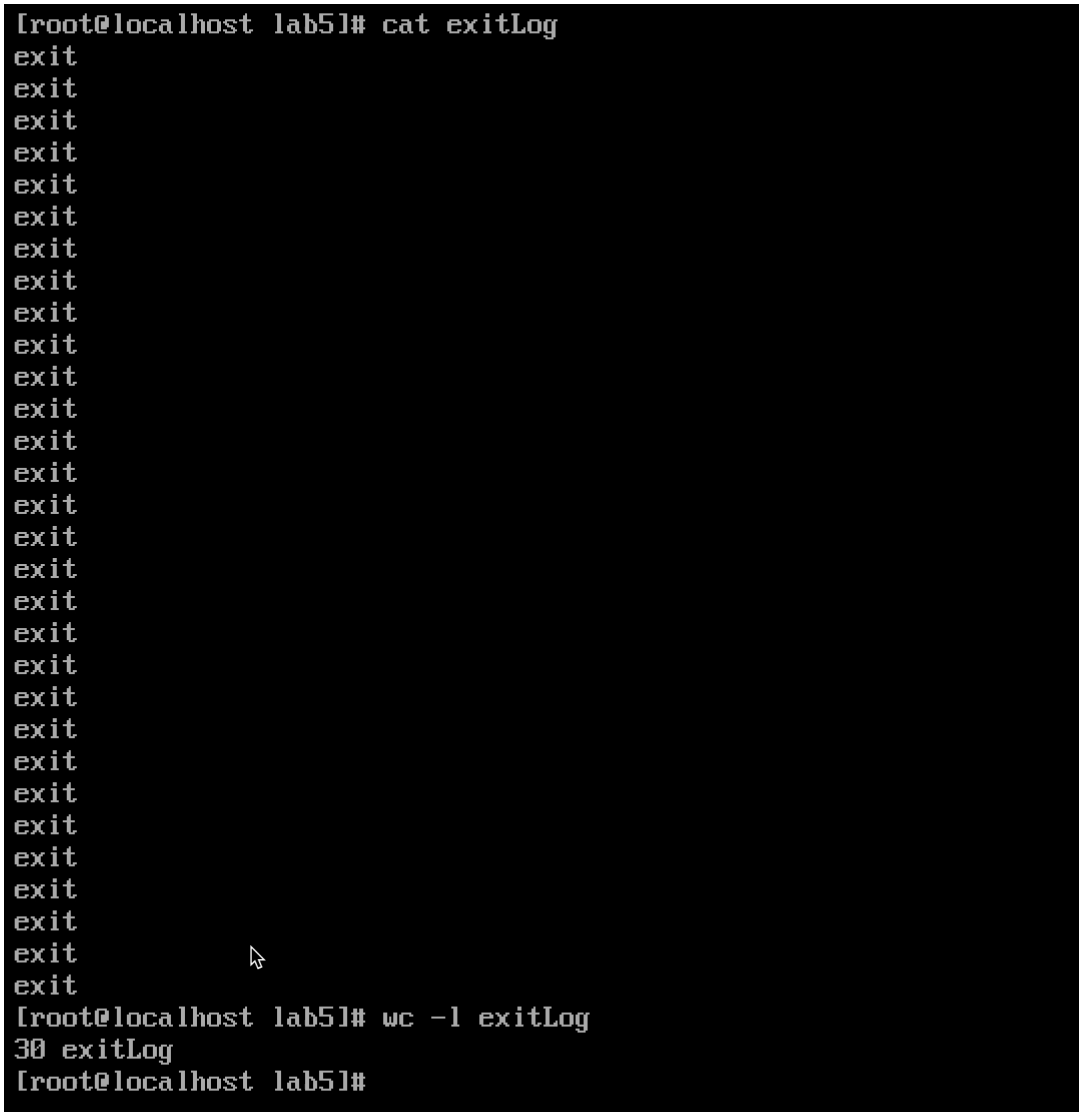


Рис. 8: **ExitLog n=700220 k=30**

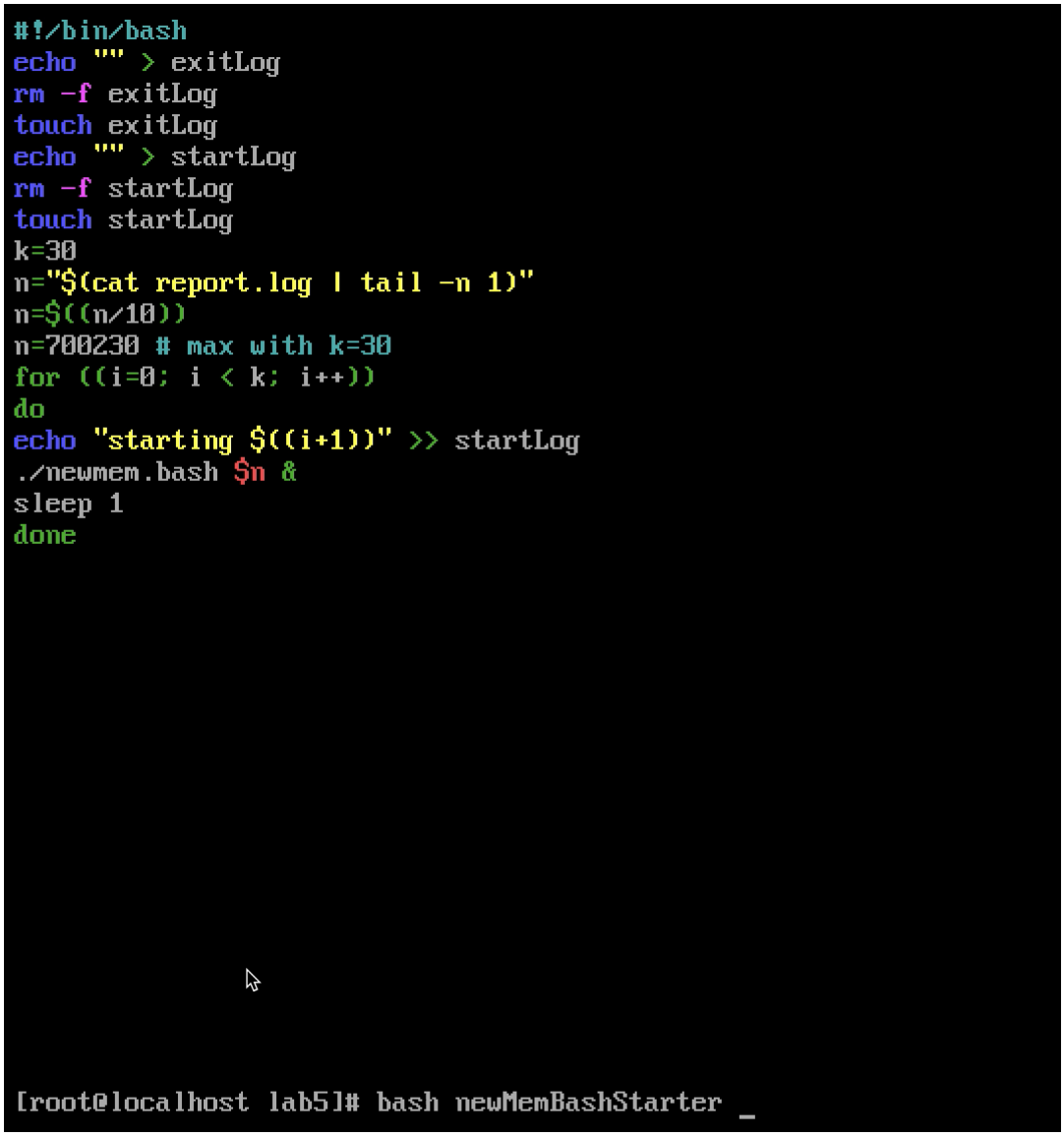


Рис. 9: **Script n=700230 k=30**

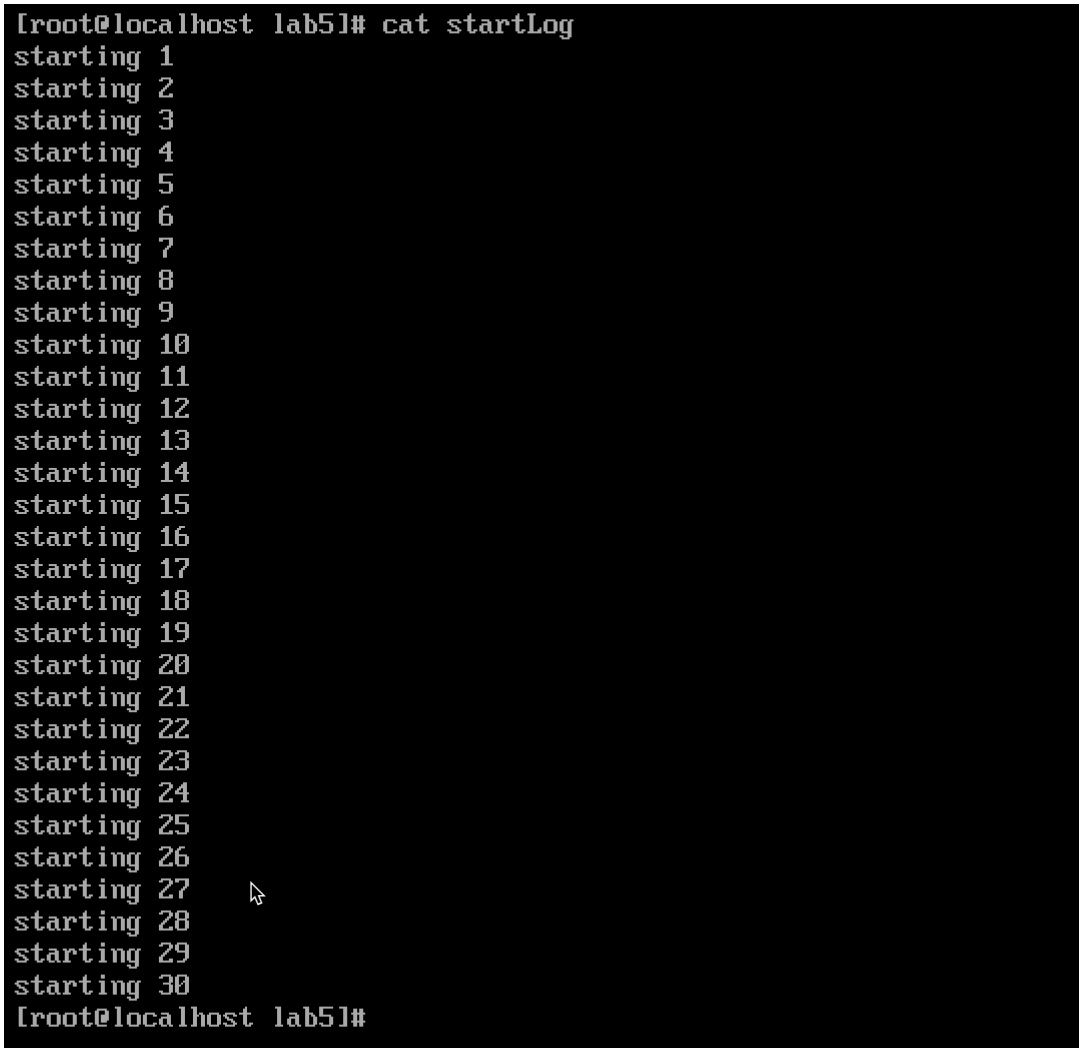


Рис. 10: **StartLog n=700230 k=30**

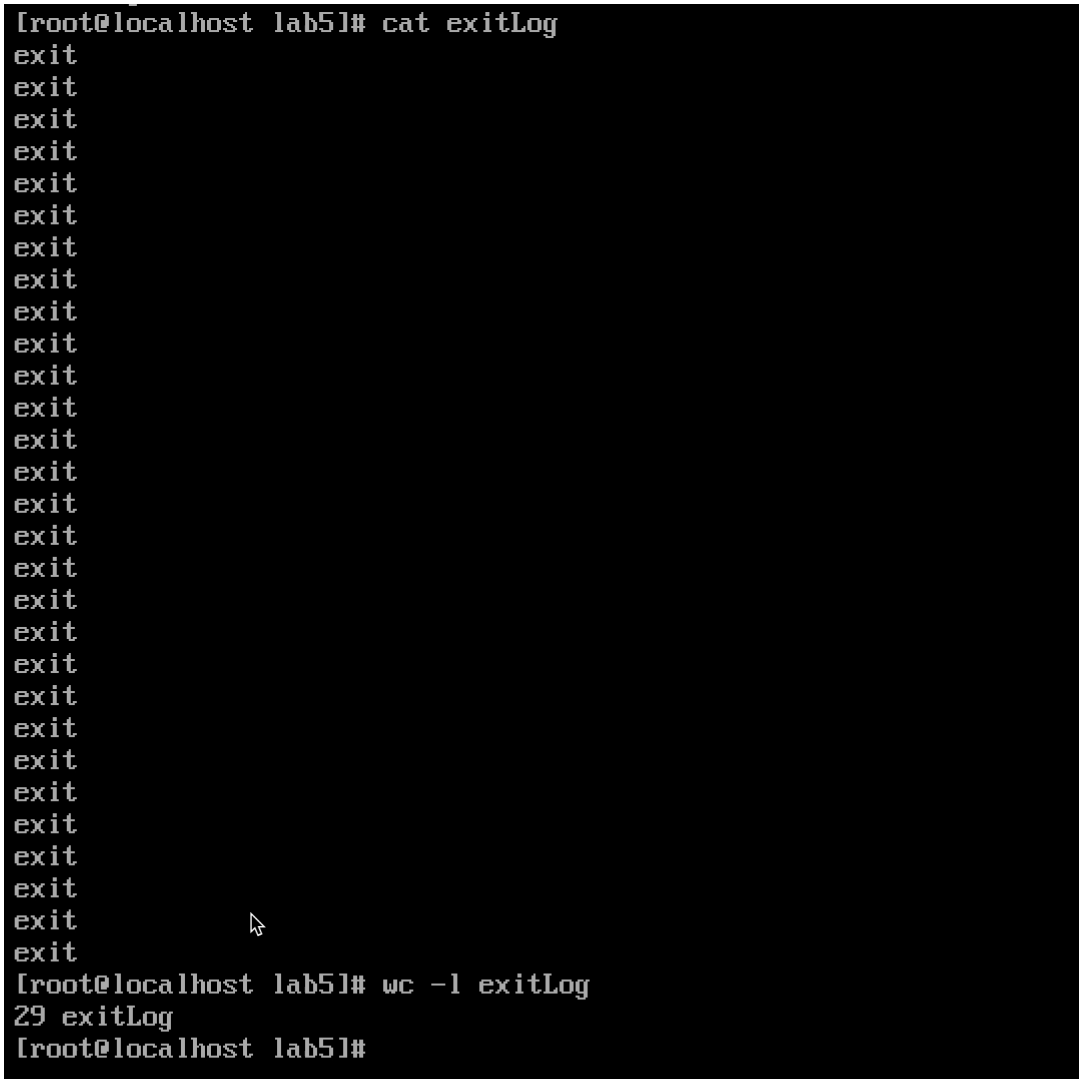
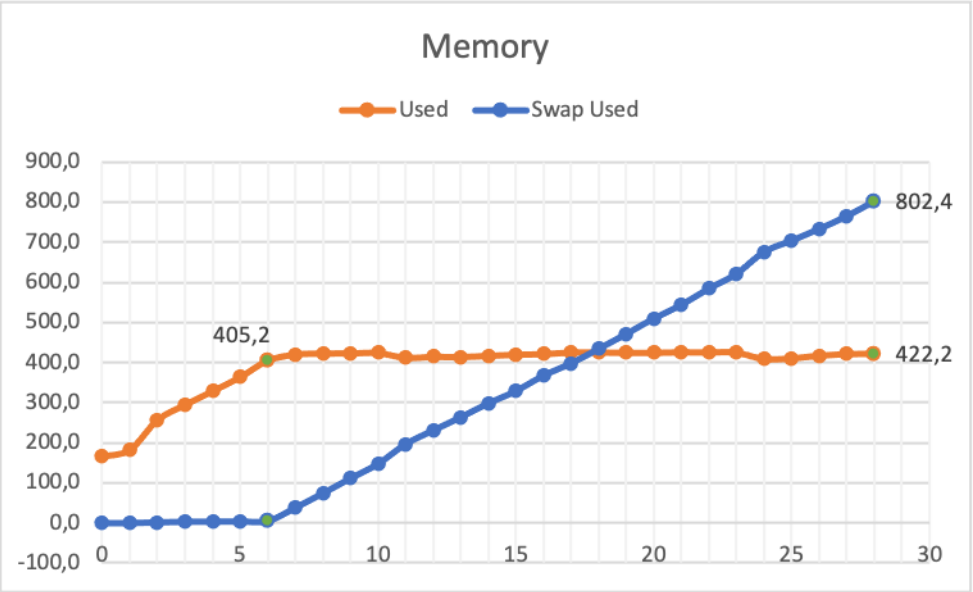


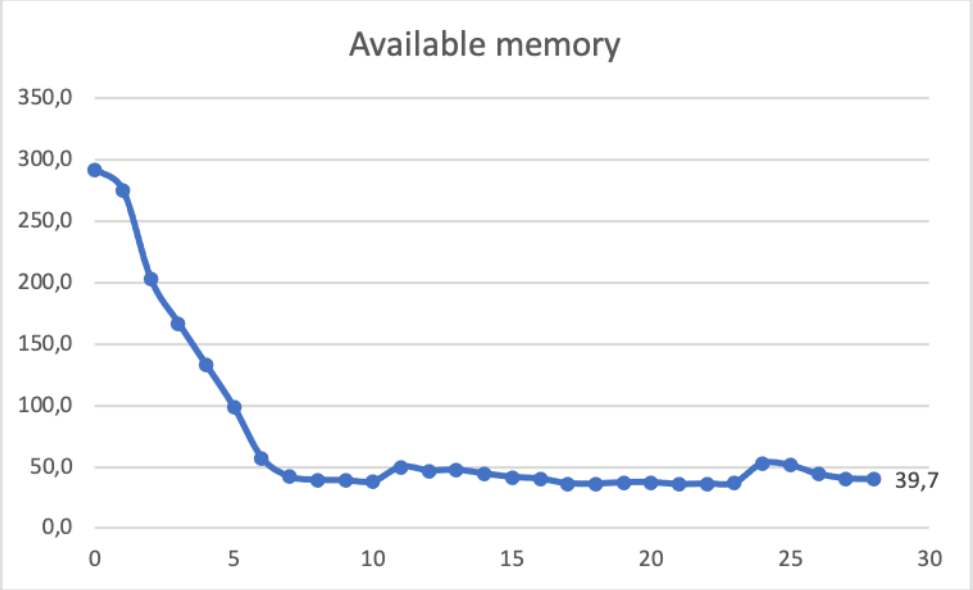
Рис. 11: **ExitLog n=700230 k=30**

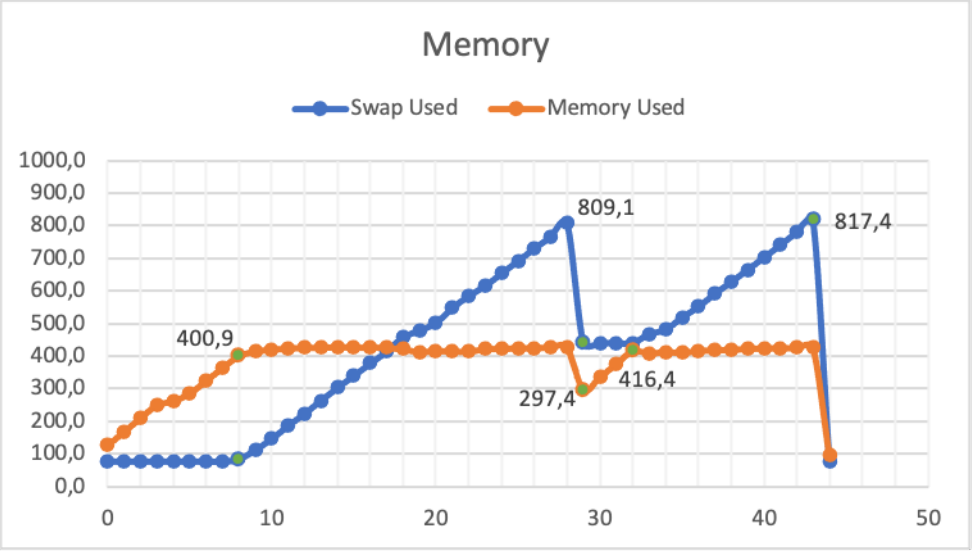
7 Обработка результатов экспериментов

Эксперимент №1

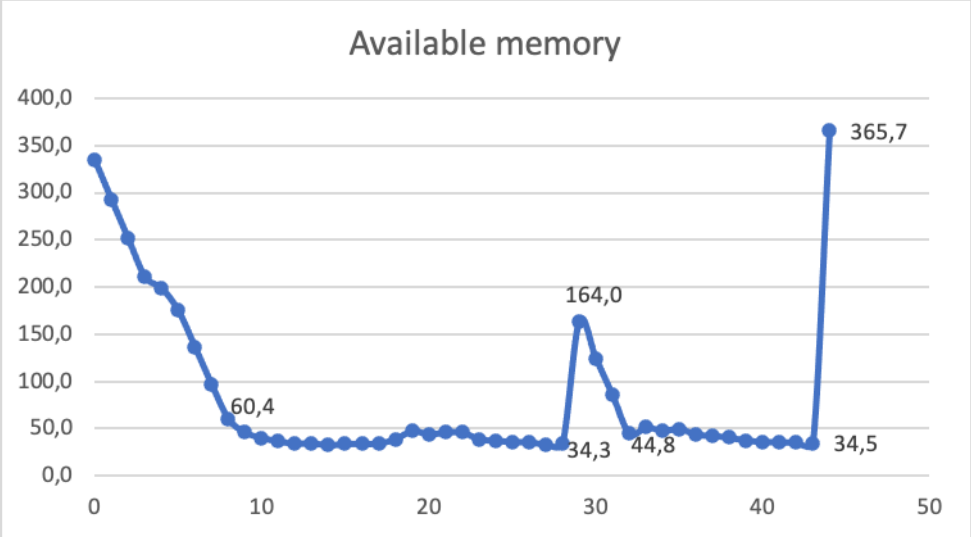
Этап №1


Memory
Used
Swap Used
900,0
800,0
700,0
600,0
500,0
400,0
300,0
200,0
100,0
0,0
-100,0
0
5
10
15
20
25
30
405,2
802,4
422,2


Available memory
350,0
300,0
250,0
200,0
150,0
100,0
50,0
0,0
0
5
10
15
20
25
30
39,7



Этап №2



8 Вывод

В результате этих двух экспериментов, мы можем увидеть:

- как происходит выделение памяти для процессов
- когда начинает использоваться память подкачки
- что происходит с памятью при завершении процесса
- в какой момент процесс заканчивается аварийно

На графиках использования памятью наглядно видно, что для обеспечения процесса памятью сначала используется физическая память и только тогда, когда в ней остается критически мало памяти, подключается подкачка, чтобы предоставить ещё больше памяти. Когда уже заканчивается и память подкачки, то процесс, который занимает там больше всех места, аварийно завершается, освобождая память для других процессов.