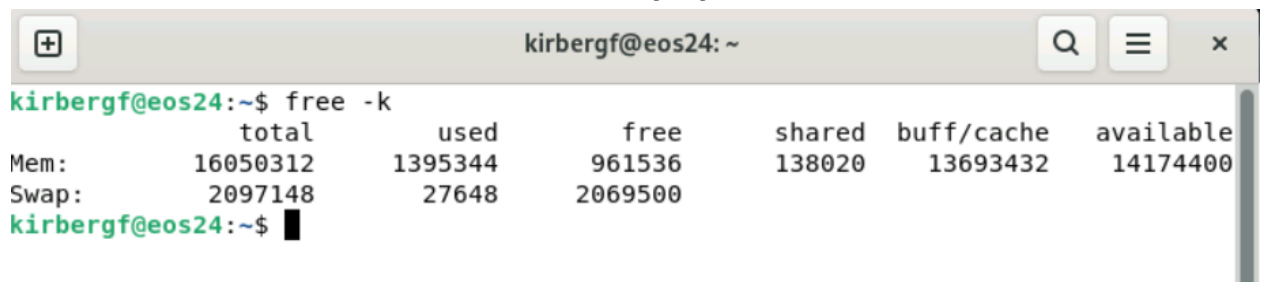


1. Determine your system configuration:

- Specify what eos system you are working on
 - use the free memory utility program to determine and report:
 - the total amount of physical memory (KB) on your system
 - the current amount of free memory (KB)
 - the total amount of swap space



```
kirbergf@eos24: ~  
kirbergf@eos24:~$ free -k  
              total        used        free      shared  buff/cache   available  
Mem:        16050312    1395344        961536      138020    13693432    14174400  
Swap:        2097148         27648       2069500  
kirbergf@eos24:~$
```

Eos system: eos24

Total amount of physical memory: 16,050,312 KB

Current amount of physical memory: 961,536 KB

Total amount of swap space: 2,097,148 KB

2. Examine and observe the memory demand of an executing process:

- Study the Sample Program
 - What is your estimate of the approximate memory demand of the Sample Program?
- Start vmstat, make your window appropriately wide, and configure it to display statistics once per second; let the system stabilize
 - Note: these experiments are best performed on a "quiet" system (i.e., not many active users)
- In another window, execute the Sample Program
 - approximately how much does the amount of free (idle) memory change?
 - considering your estimated memory demand of the Sample Program (question 2a), explain why the observed change is an expected result.

The image shows two terminal windows. The left window shows the execution of `sampleProgramOne`. The right window shows the output of the `vmstat 1` command, which displays system statistics every second. The output includes columns for processes, memory, swap, IO, system, and CPU usage. The 7th line of the output shows a significant decrease in free memory (from 927416 to 926260 bytes) and an increase in used memory (from 13044504 to 13044508 bytes) when the sample program is executed.

```
kirbergf@eos24: ~/cis452/lab10
kirbergf@eos24:~/cis452/lab10$ ./sampleProgramOne
kirbergf@eos24:~/cis452/lab10$

kirbergf@eos24:~/cis452/lab10$ vmstat 1
procs-----memory-----swap-----io-----system-----cpu-----
r  b   swpd   free   buff   cache   si   so    bi    bo    in   cs  us  sy  id  wa  st
0  0   27648  927416  665180 13044504 0    0     2     4     6    1  0  0  99  0  0
0  0   27648  928468  665180 13044504 0    0     0    0    452 1164 1  0  99  0  0
0  0   27648  927264  665184 13044504 0    0     0    0    44 1878 7279 10 1  89  0  0
0  0   27648  927296  665184 13044508 0    0     0    0   104 607 1721 1  0  98  0  0
1  0   27648  927376  665184 13044508 0    0     0    0    339 1214 0  0  99  0  0
0  0   27648  926776  665184 13044512 0    0     0    0    392 800 1  0  99  0  0
1  0   27648  910144  665184 13044508 0    0     0    0    569 1097 7  0  93  0  0
1  0   27648  911268  665184 13044508 0    0     0    0    905 1422 9  0  91  0  0
0  0   27648  928288  665184 13044508 0    0     0    0    48 453 1309 1  0  99  0  0
1  0   27648  927416  665184 13044512 0    0     0    0   1897 7173 12 1  87  0  0
0  0   27648  926260  665184 13044508 0    0     0    0    498 1274 1  0  99  0  0
0  0   27648  926500  665184 13044508 0    0     0    0    424 1085 1  0  99  0  0
^C
kirbergf@eos24:~/cis452/lab10$
```

Estimate of the approximate memory demand:

`intPtr` is malloced (`dim * dim * sizeof(int)`) and `dim = COEFFICIENT * KB`.

`COEFFICIENT = 2`, `KB = 1024`, and `sizeof(int) = 4`

Therefore, the approximate memory demand is $2048 * 2048 * 4 = 16,777,216$ bytes.

Amount of free memory change:

16,632 KB as on the 7th line the sample program is executed and the amount of free memory is reduced by 16,632 KB.

Observer change as expected result:

As 16,777,216 Bytes is equal to 16,384 KB, the observed change is expected as it is approximately the amount of memory that `sampleProgramOne` was expected to use for its execution.

3. Examine the effect of memory access patterns:

- Read the man pages for the time utility program. Then use `/usr/bin/time` together with command-line arguments as described for time to obtain complete statistics (i.e., run in verbose mode). Execute and time the Sample Program.

- obtain basic statistics

- what is the size of a page in Linux?
- how long does the program take to run?

- Change the memory access statement in the Sample Program to read:

```
intPtr[j * dim + i] = (i + j) % count;
```

then re-compile and re-run the program, collecting statistics on execution time.

- Precisely, how does this change alter the program's memory access pattern (i.e. what memory objects get "touched", and in what order)? A diagram will help here.
- How does this change affect the program's (user) execution time?
- Precisely, why does the change have the observed effect (your answer must incorporate an important concept related to paging and virtual memory)?

Before change:

```
kirbergf@eos24: ~/cis452/lab10
kirbergf@eos24:~/cis452/Lab10$ /usr/bin/time -v ./sampleProgramOne
Command being timed: "./sampleProgramOne"
User time (seconds): 1.85
System time (seconds): 0.00
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:01.86
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 17408
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 4162
Voluntary context switches: 4
Involuntary context switches: 4
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
kirbergf@eos24:~/cis452/Lab10$
```

After change:

```
kirbergf@eos24: ~/cis452/lab10
kirbergf@eos24:~/cis452/Lab10$ /usr/bin/time -v ./sampleProgramOne
Command being timed: "./sampleProgramOne"
User time (seconds): 4.77
System time (seconds): 0.01
Percent of CPU this job got: 99%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:04.79
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 17408
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 1
Minor (reclaiming a frame) page faults: 4162
Voluntary context switches: 5
Involuntary context switches: 11
Swaps: 0
File system inputs: 32
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
kirbergf@eos24:~/cis452/Lab10$
```

Size of page: 4096 bytes

Run time before change: 1.85 seconds

Change in memory access pattern and diagram:

After the change, the memory objects now get "touched" column by column ([0,0], [1,0], [2,0], etc.), whereas previously it was row by row ([0,0], [0,1], [0,2], etc.).

Change in execution time: Increased to 4.77 seconds

Change as the observed effect:

The change has this effect because having the memory accessed column by column results in memory being accessed “out of order”/in an unsorted manner. This affects the concept of caching, as there is less spatial locality, resulting in a longer execution time.

4. Examine the use of virtual memory:

- Perform the following
 - Change the memory access pattern for the Sample Program back to its original form
 - Change the LOOP value to 1 (or be prepared to wait a long time)
- Adjust the COEFFICIENT parameter in the Sample Program to a value that causes the memory demand of the program to exceed the total amount of physical memory on your machine (as determined in question 1 above). Note: memory demand should exceed the amount of RAM on your system, but must be less than (RAM + Swap).
 - What value did you use, justify your computation
- Configure and run vmstat to display statistics once every second and use /usr/bin/time in verbose mode to execute and time the program
 - Observe vmstat system statistics as the program executes. What happens to the amount of free memory (during and after the run)? Describe all the other fields that have changed (including non-memory fields), and describe why they have changed.
 - Explain how the operating system is adapting to the increased memory demand of the Sample Program. Include a brief discussion of the execution time and the number of page faults incurred. Your explanation should demonstrate that you understand what is happening from a virtual memory viewpoint.

The screenshot shows two terminal windows. The left window displays the output of `/usr/bin/time -v ./sampleProgramOne`, showing a command being timed, user time of 9.55s, system time of 4.18s, 90% CPU usage, and a total elapsed time of 15.19s. It also lists various system statistics like shared text size, unshared data size, stack size, total size, resident set size, page faults, context switches, and file/socket messages. The right window shows the output of `vmstat 1`, displaying a table of system statistics including processes, memory, swap, IO, system, and CPU usage over time.

```
kirbergf@eos24: ~/cis452/lab10
kirbergf@eos24:~/cis452/Lab10$ /usr/bin/time -v ./sampleProgramOne
Command being timed: "./sampleProgramOne"
User time (seconds): 9.55
System time (seconds): 4.18
Percent of CPU this job got: 90%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:15.19
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 15106688
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 13
Minor (reclaiming a frame) page faults: 4064330
Voluntary context switches: 379
Involuntary context switches: 822
Swaps: 0
File system inputs: 192
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0

kirbergf@eos24:~/cis452/Lab10$ vmstat 1
procs -----memory-----swap-----io-----system-----cpu-----
r b swpd free buff cache si so bi bo in cs us sy id wa st
0 0 820608 15021756 11692 343756 0 1 2 5 0 2 0 0 99 0 0
0 0 820608 15021504 11692 343756 0 0 68 0 301 884 0 0 99 0 0
0 0 820608 15020844 11692 343824 24 0 24 0 1838 6971 8 1 90 1 0
1 0 820608 14082916 11700 343836 0 0 0 44 750 1447 6 2 93 0 0
2 0 820608 12844016 11700 343836 20 0 20 0 2024 6793 11 3 86 0 0
1 0 820608 11551004 11700 343836 0 0 0 0 767 1277 6 2 91 0 0
1 0 820608 10253580 11700 343836 0 0 0 0 777 943 7 2 91 0 0
1 0 820608 8957796 11700 343836 0 0 0 92 630 981 6 2 91 0 0
1 0 820608 7662768 11708 343836 8 0 8 44 732 882 7 3 91 0 0
1 0 820608 6364968 11708 343836 0 0 0 0 517 834 6 2 92 0 0
1 0 820608 5070940 11708 343836 8 0 8 0 766 937 7 2 91 0 0
1 0 820608 3773896 11708 343888 0 0 52 0 679 724 7 2 91 0 0
1 0 820608 2474492 11708 343888 8 0 8 0 844 1066 7 2 90 0 0
2 1 820608 1180124 11712 344236 0 0 352 40 672 836 6 2 91 0 0
5 0 821888 145664 11424 263416 0 22108 0 22112 4558 2291 6 6 88 1 0
0 3 1154432 151920 396 151072 32 332424 4044 332424 39444 2953 2 8 80 10 0
0 4 1591732 149916 696 161328 0 437248 10656 437392 59808 2493 3 3 74 19 0
1 3 2032708 155504 1128 165216 264 444980 5704 444980 70802 2428 3 3 71 23 0
0 0 845712 15254396 4528 177540 428 50496 16096 50496 4315 2852 1 3 92 4 0
0 0 845712 15253388 4536 178024 0 0 728 36 404 1011 1 0 99 0 0
0 0 845712 15253568 4536 178264 64 0 64 0 459 913 1 0 99 0 0
```

Value:

COEFFICIENT = 63

As the memory demand must exceed the amount of RAM (16050312 KB) but must also be less than the RAM + Swap (16050312 + 2097148 = 18147460 KB).

Using a coefficient of 64 the memory demand of the program is $((1024 * 63) * (1024 * 63) * \text{sizeof(int)}) = 16647192576$ bytes or 16257024 KB.

Changes:

The amount of free memory gradually decreases during the program execution. After execution the amount of free memory returns to approximately the initial values.

ID	Description	Why it changed
r	# of processes in a running state	Has a sharp increase right before virtual memory used increases because memory is being managed to accommodate the processes
b	# of processes in uninterruptible sleep state	Has a sharp increase during virtual memory usage as these processes are likely related to I/O that are waiting for swapping to finish
swpd	Amount of virtual memory used (swap space)	Has a sharp increase as free/idle memory runs low during program execution
free	Amount of idle memory	Gradually decreases during program execution
buff	Amount of memory used as buffers	Decreases during virtual memory usage as it is being utilized to temporarily hold data
cache	Amount of memory used as cache	Decreases during virtual memory usage as pages are being stored on the cache
si	Amount of memory swapped in from disk	Increases during virtual memory usage as pages are swapped in to physical memory
so	Amount of memory swapped to disk	Increases during virtual memory usage as pages are swapped out

		of physical memory
bi	Blocks received from a block device	Increases during virtual memory usage as blocks are read from disk
bo	Blocks sent to a block device	Increases during virtual memory usage as blocks are written to disk
in	# of interrupts per second	Increased I/O operations used for swapping result in an increase of interrupts
cs	# of context switches per second	Increased I/O operations used for swapping result in an increase of context switches to maximize CPU usage while waiting for I/O operations
us	Time spent running non-kernel code	Increases during swapping due to an increase in tasks
sy	Time spent running kernel code	No substantial changes
id	Time spent idle	No substantial changes
wa	Time spent waiting for I/O	Swapping involves I/O operations which results in increases I/O wait times
st	Time stolen from a virtual machine	No substantial changes