Part 1: Activity Monitor

1.

The processes that use the most virtual memory (VM) are listed as follows:

* Firefox (394.8 MB VM, 432.2 MB RM
* LabStatsClient\_pager (353.2 MB VM, 36.1 MB RM)
* PaperCut NG (253.2 MB VM, 44.2 MB RM)
* mds (220.7 MB VM, 82.9 MB RM)
* Microsoft Word (183.2 MB VM, 123.8 MB RM)

3 out of 5 of the above processes are in the bottom 5 for Real Memory use (namely Firefox, Microsoft Word, and mds)

2.

The processes that use the least virtual memory (VM) are listed as follows:

* bash (not sure why AM lists 2; I only have one terminal open)

(9.4 MB VM, 724 KB RM)

* dynamic\_pager (17.4 MB VM, 756 KB RM)
* cron (18.6 MB VM, 836 KB RM)
* zsh (18.6 MB VM, 520 KB RM)
* ntpd (18.6 MB VM, 1.4 MB RM)

4 out of 5 of the above processes are in the bottom 5 for Real Memory use (all of them except for ntpd are).

3.

The System Memory at the bottom lists 204.22 GB swap space (labeled VM size). This hard drive has 356.18 GB (of 500 GB) of free space available.

4.

Page ins: 1,013.6 MB

Page outs: 1.09 GB

Swap used: 10.3 MB

The memory management system seems to be working quite effectively. Regarding swaps, only 10.3 MB of swaps are used, which is quite insignificant in comparison to the 12 GB of main memory this machine has, most of the memory occupied by processes are likely to not be disturbed by swaps throughout their lifetime. This is good because swaps are generally more expensive than page ins and page outs.

With approximately 1 GB of page ins and page outs, the virtual memory does not seem to have a high demand either, especially in comparison to the amount of main memory this computer has (12 GB). In this case, I don’t think adding main memory would make the computer much faster, if at all, as the amount of paging and swapping the main memory uses is not the bottleneck for the computer’s overall speed.

Part 2: vm\_stat

5.

Page size of 4096 bytes

6.

Pages free: 2256371

Pages active: 332099

Pages inactive: 82609

Pages speculative: 154975

Pages wired down: 318473

7.

*According to Activity Monitor:*

9.15 GB of memory is free

1.30 GB of memory is active

348.1 MB of memory is inactive

1.22 GB of memory is wired

2.85 GB of memory is used

8.

The “used memory” category must include the “speculative” category in vm\_stat.

­

Part 3: swapping

9.

Before free memory runs out, real memory usage just keeps going up. After free memory runs out, the real memory usage starts decreasing, such that free real memory increases.

10.

Before free memory runs out, virtual memory stays slightly above real memory usage. After free memory runs out, the virtual memory usage keeps increasing, but the real memory usage decreases, such that the difference between virtual memory and real memory usage increments.

This is possibly because virtual memory is required to continually make space for the program to run even after all free memory is occupied, and it does this by using paging techniques to reference parts of memory that gets replaced by address spaces. Virtual memory in this case shows how much memory the program is actually using, and as I keep iterating through the loop, eventually the memory used by the python program exceeds the amount of main memory this computer has.

11.

Before and after free memory runs out, % CPU usage goes up to 100% as the program is running and the counter is incrementing.

The reason it behaves this way is probably because the CPU operates separately from the operating system, which does all the memory allocation, such that the amount of available free / used memory does not affect the way the CPU performs.

12.

Before free memory runs out, the amount of page ins and page outs stays the same and are about equal. After free memory runs out, the amount of page outs increases, whilst the amount of page ins stays constant.

This is because page in is transferring a page from the disk to RAM, whereas page out is transferring a page from RAM to disk. As we are never reading from the disk, but simply using more and more main memory, it makes sense that the amount of page outs increase as the amount of page ins remain constant.

13.

After free memory runs out, the number of “swap used” (which was initially 10.3 MB) increases fairly quickly (up to 6 GB). However, after I exit the program, the amount of swap used goes back down, yet it is still substantially larger than the initial 10.3 MB, being 541.5 MB.

14.

*Initial values:*

page ins: 1.14 GB

page outs: 2.64 GB

*Final values:*

page ins: 1.32 GB

page outs: 3.52 GB

It turns out that test 2 differs from test 1 in that both page ins and page outs increase after all free memory gets used up. This is because in the code, test 2 “adds at a place relatively far from last read”, which implies that a brief calculation has to be done to calculate the new disk address. This requires reading the old disk address and incrementing the counter. Obviously, this does not take much time, so the amount of page ins increments a lot lower than the amount of page outs.