**1.The first Linux tool you should check out is the very simple tool free. First, type man free and read its entire manual page; it’s short, don’t worry!**

Display total amount of free and used physical and swap memory in the system, as well as the buffers and caches used by the kernel.

**2.Now, run free, perhaps using some of the arguments that might be useful (e.g., -m, to display memory totals in megabytes). How much memory is in your system? How much is free? Do these numbers match your intuition?**

There are 7925 megabytes memory in my Linux machine, and 4336 megabytes is free. Yes, this numbers match my intuition.

**4.Now, while running your memory-user program, also (in a different terminal window, but on the same machine) run the free tool. How do the memory usage totals change when your program is running? How about when you kill the memory-user program? Do the numbers match your expectations? Try this for different amounts of memory usage. What happens when you use really large amounts of memory?**

When I run this program, with command: **./memory-user.c 10,** total free memory become 4326. Next we kill the process the free memory become 4363 again. We also try the **./memory-user.c 100** , it becomes 4263, and when we kill it, free memory become 4363 again. When use **3000** megabyts, the terminal show **‘malloc filed’**

**5. Let’s try one more tool, known as pmap. Spend some time and read the pmap manual page in detail.**

The pmap command reports the memory map of a process.

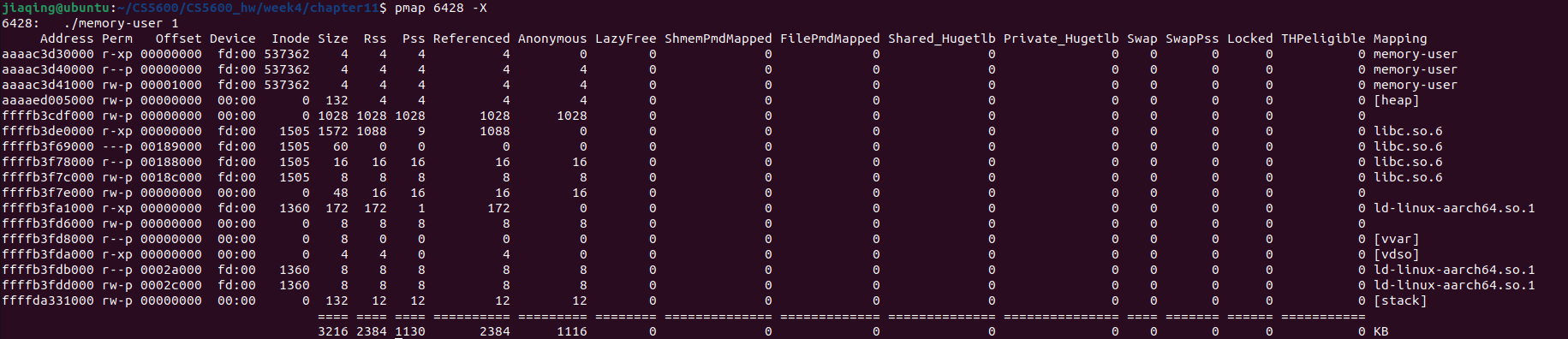
**6. To use pmap, you have to know the process ID of the process you’re interested in. Thus, first run ps auxw to see a list of all processes; then, pick an interesting one, such as a browser. You can also use your memory-user program in this case (indeed, you can even have that program call getpid() and print out its PID for your convenience).**

****

**Text

Description automatically generated**

**7. Now run pmap on some of these processes, using various flags (like -X) to reveal many details about the process. What do you see? How many different entities make up a modern address space, as opposed to our simple conception of code/stack/heap?**



I see first three rows are the program code locate at address aaaac3d30000 to aaaaed005000.We also can see heap start at aaaaed005000. The stack start at bottom ffffda331000

**8. Finally, let’s run pmap on your memory-user program, with different amounts of used memory. What do you see here? Does the output from pmap match your expectations?**

With command: **./memory-user 1, ./memory-user 2, ./memory-user 10**

We see the memory in heap with 1 megabytes is 1028 bytes in pmap pid -X, with 2 megabytes the heap size growth to 2052 bytes, when we set to 10 megabytes, heap growth to 10244 bytes. Basically match my expectations but couple of bytes off.