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).

```
jiaqing@ubuntu:~/CS5600/CS5600_hw/week4/chapter11$ ./memory-user 1
Current process id is 6428
```

```
jiaqing@ubuntu:~/CS5600/CS5600_hw/week4/chapter11$ pmap 6428
6428:   ./memory-user 1
0000aaaac3d30000      4K r-x-- memory-user
0000aaaac3d40000      4K r---- memory-user
0000aaaac3d41000      4K rw--- memory-user
0000aaaaed005000    132K rw---   [ anon ]
0000ffffb3cdf000   1028K rw---   [ anon ]
0000ffffb3de0000   1572K r-x-- libc.so.6
0000ffffb3f69000     60K ----- libc.so.6
0000ffffb3f78000     16K r---- libc.so.6
0000ffffb3f7c000      8K rw--- libc.so.6
0000ffffb3f7e000     48K rw---   [ anon ]
0000ffffb3fa1000    172K r-x-- ld-linux-aarch64.so.1
0000ffffb3fd6000      8K rw---   [ anon ]
0000ffffb3fd8000      8K r----   [ anon ]
0000ffffb3fda000      4K r-x--   [ anon ]
0000ffffb3fdb000      8K r---- ld-linux-aarch64.so.1
0000ffffb3fdd000      8K rw--- ld-linux-aarch64.so.1
0000ffffda331000    132K rw---   [ stack ]
 total             3216K
```

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?

```
jiaqing@ubuntu:~/CS5600/CS5600_hw/week4/chapter11$ pmap 6428 -X
6428:   ./memory-user 1
         Address Perm   Offset Device  Inode Size  Rss  Pss Referenced Anonymous LazyFree ShmemPmdMapped FilePmdMapped Shared_Hugetlb Private_Hugetlb Swap SwapPss Locked THPeligible Mapping
aaaac3d30000 r-xp 00000000  fd:00 537362    4    4    4          4         0        0              0             0              0               0    0       0      0           0 memory-user
aaaac3d40000 r--p 00000000  fd:00 537362    4    4    4          4         4        0              0             0              0               0    0       0      0           0 memory-user
aaaac3d41000 rw-p 00001000  fd:00 537362    4    4    4          4         4        0              0             0              0               0    0       0      0           0 memory-user
aaaaed005000 rw-p 00000000  00:00      0  132    4    4          4         4        0              0             0              0               0    0       0      0           0 [heap]
ffffb3cdf000 rw-p 00000000  00:00      0 1028 1028 1028       1028      1028        0              0             0              0               0    0       0      0           0
ffffb3de0000 r-xp 00000000  fd:00   1505 1572 1088    9       1088         0        0              0             0              0               0    0       0      0           0 libc.so.6
ffffb3f69000 ---p 00189000  fd:00   1505   60    0    0          0         0        0              0             0              0               0    0       0      0           0 libc.so.6
ffffb3f78000 r--p 00188000  fd:00   1505   16   16   16         16        16        0              0             0              0               0    0       0      0           0 libc.so.6
ffffb3f7c000 rw-p 0018c000  fd:00   1505    8    8    8          8         8        0              0             0              0               0    0       0      0           0 libc.so.6
ffffb3f7e000 rw-p 00000000  00:00      0   48   16   16         16        16        0              0             0              0               0    0       0      0           0
ffffb3fa1000 r-xp 00000000  fd:00   1360  172  172    1        172         0        0              0             0              0               0    0       0      0           0 ld-linux-aarch64.so.1
ffffb3fd6000 rw-p 00000000  00:00      0    8    8    8          8         8        0              0             0              0               0    0       0      0           0
ffffb3fd8000 r--p 00000000  00:00      0    8    0    0          0         0        0              0             0              0               0    0       0      0           0 [vvar]
ffffb3fda000 r-xp 00000000  00:00      0    4    4    0          4         0        0              0             0              0               0    0       0      0           0 [vdso]
ffffb3fdb000 r--p 0002a000  fd:00   1360    8    8    8          8         8        0              0             0              0               0    0       0      0           0 ld-linux-aarch64.so.1
ffffb3fdd000 rw-p 0002c000  fd:00   1360    8    8    8          8         8        0              0             0              0               0    0       0      0           0 ld-linux-aarch64.so.1
ffffda331000 rw-p 00000000  00:00      0  132   12   12         12        12        0              0             0              0               0    0       0      0           0 [stack]
                                       ==== ==== ==== ========== ========= ======== ============== ============= ============== =============== ==== ======= ====== ==========
                                       3216 2384 1130       2384      1116        0              0             0              0               0    0       0      0           0 KB
```

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.