Assignment: free and pmap

Use this form to complete your assignment. Students may work alone or with a partner. Do the homework set at the end of Chapter 13. To do this activity you will need to use a Linux system such as linux.cs.pdx.edu so that you can run the tools "free" and "pmap". Mac and Windows systems, for example, do not support these tools, so you need to work on a linux system.

Your email address (**hermann@pdx.edu**) will be recorded when you submit this form. Not you? Switch account

* Required

Your Name *

Hermann Yepdjio

Partner's Name (if you worked with a partner)

N.A

Read the man page for the free tool and run "which free" to make sure that you have access to the free tool. *

✓ check here after you have studied the man page for the 'free' command

Start by running "free --giga -s 1 -c 10" and then run free with other switch combinations to get a feel for what the command does. How much memory is in your system? How much is free? *

24 GB in my system and 15 GB is free

Next, build and run the program mem.c found in the vm-beyondphys directory within the OSTEP homework directories. The mem program takes one command-line argument: the number of megabytes of memory it will use. When run, it allocates that amount of memory and initializes an array of integers within that memory region. Then it repeatedly streams through the array, incrementing each element. The program runs indefinitely until you stop it. *

✓ check here after you have created and tested the mem program

Now, while running your mem 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 after you stop the mem program? Do the numbers match your expectations? Try this multiple times and vary the amounts of memory usage. What happens when you use large amounts of memory (50% of your system's amount of free memory or more)? *

I ran the program passing 1024 (1GB) as command-line argument and then run the free command on another terminal and I noticed that the free memory decreased from 15 GB to 14 GB. When I stopped the mem.c program, the free memory went back to 15 GB. So, yes! The numbers match my expectations.
I observed the same behavior when using half of the total free memory.

Next, study the tool known as pmap. Spend some time, and read the pmap manual page in detail. *

✓ check here after you have studied the man page for the 'pmap' command

To use pmap, you must know the process ID of the process you're interested in. Run your mem program in a separate terminal window and note its pid, either by modifying it to call the system call getpid() and printing the result or by using a command such as "ps aux" or "top" in a separate window to obtain the pid for your mem program. *

✓ Done. I am able to run my mem program and obtain its pid

Now run pmap on your mem program, 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 the simple model of code/stack/heap described in the lecture and book? *

I get a page table with 19 lines of information, each corresponding to an entity of the address space. For each entity, I get information such as its address, permission access, offset, size, mapping info, etc...
I also get the total size of the page table
Since I have 19 different addresses, I suppose a modern address space has 19 different entities.

Run your mem program several times with widely varying memory allocation inputs. run pmap on your mem program each time. What do you see here? Does the output from pmap match your expectations? *

I get a page table which provides information about its memory blocks. Its total size is about the same as the argument I passed when running mem.c. So, yes! The output of pmap matches my expectations.

Submit