CSC 369 Worksheet 6 Solution

August 21, 2020

1. Done. See link here

2. First, I need to find out how much memory is in my system, and how much is free. Running the command `free -m`, we have

```
ubuntu@primary:~$ free -m
              total        used        free      shared  buff/cache   available
Mem:            981         157         131           0         691         668
Swap:             0           0           0
```

   Using this information, I can write that the computer has

   - 981 MB of total memory
   - 131 MB of free memory

   Second, I need to answer if these numbers match my intuition.

   It does match my inution that free memory should be less than total memory.

   Notes

   - I should ask professor the type of intution the author was expecting
   - Installing Ubuntu Virtual Machine link: here
   - Start virtual machine by typing command: `multipass start ubuntu-lts`

3. I need to write a little program `question_3.c` (I will create this instead of `memory-user.c` for recording keeping purposes) that uses a certain amount of memory.

   It's critera are:

- The program should take one command-line argument: the number of megabytes of memory it will use.
- The program should allocate an array.
- The program should constantly stream through the array, touching each entry.
- The program should do this indefinitely, or for a certain amount of time also specified in command line.

For it's solution, please referr to `question_3.c`

Notes

- Learned that a large array (> 5000) can be generated using heap memory [1]
- **Command Line Arguements in C**

```
#include <stdio.h>

int main (int argc, char *argv[])
{

  return 0;
}
```

  - `argc`
    - Is the number of arguments passed to the program
  - `argv`
    - Is an array of strings
    - Each string represents one of the arguements passed to the program

  Example

For example, the command line

```
gcc -o myprog myprog.c
```

would result in the following values internal to GCC:

```
argc
     4
argv[0]
     gcc
argv[1]
     -o
argv[2]
     myprog
argv[3]
     myprog.c
```

- **String to integer**
  - **Syntax:** `int atoi(const char *string)`

    Example

    ```
    int output = atoi("20") /* Stores 20 in output*/
    ```

References

1) Stackoverflow, Allocating A Large (5000+) Array, link
2) Techdelight.com. Hpw tp fomd execution time of a C program, link
3) Stackoverflow, How do you clear the console screen in C, link

4. First, I need to revise program made in question 3 to also run the `free` tool.

   For it's solution, please referr to `question_4.c`.

   Second, I need to answer how the memory is changing when the programming is running.

   As program runs, the amount of memory available for the program decreases.

   Once it runs out, it returns "Resource temporarily unavailable" error.

   Third, I need to answer what happens to memory when program is killed.

   In the current solution, I am using heap memory, and I haven't set it up for the array to be freed on kill signal. So, the lost memory is not reclaimed once killed.

Fourth, I need to answer what happens when a large amount of memory is used.

The answer is the same as second part of the question. It tries to fill up, but once available memory runs out, it force terminates with "Resource unavailable" error.

```
Current index: 14
Time elapsed: 14 seconds
Amount of memory allocated: 10 MB
Size of array: 2500000
Press ctrl + z to terminate program
vm.swapusage: total = 1024.00M  used = 274.00M  free = 750.00M  (encrypted)
ERROR: Fork failed: Resource temporarily unavailable
moegu@MacBook-Pro-5 worksheet_6 % sysctl vm.swapusage
vm.swapusage: total = 1024.00M  used = 421.00M  free = 603.00M  (encrypted)
```

Notes

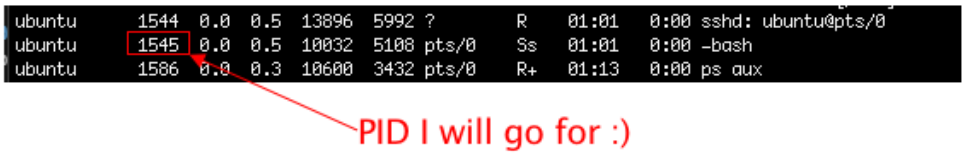
- Mac's equivalent of Ubuntu's `free -m` is `sysctl vm.swapusage` [1]

References

1) Stack Overflow, Is there a Mac OS X Terminal version of the "free" command in Linux systems? ,link

5. Please see link here

6.



Notes

- Alternative of pmap on Mac is `vmmap <PID>` [1]

References

1) Yong Sun's Blog, Tips: the equivalents of ldd(1) and pmap(1) on Mac OS X, link

7. First, I need to answer the question of what I see when running the command `pmap` with flags like `-X`

On Running the command, what I see is rows of virtual memory address and columns and of information regarding the virtual memory address.

```
ubuntu@primary:~$ pmap -X 1545
1545:   -bash
         Address Perm   Offset Device Inode  Size  Rss  Pss Referenced Anonymous LazyFree ShmemPmdMapped FilePmdMapped Shared_Hugetlb Private_Hugetlb Swap SwapPss Locked THPeligible Mapping
    55a3c0f53000 r--p 00000000 fc:01  1494   180  180  180        180         0        0               0             0              0               0    0       0      0           0 bash
    55a3c0f80000 r-xp 0002d000 fc:01  1494   708  708  708        708         0        0               0             0              0               0    0       0      0           0 bash
    55a3c1031000 r--p 000de000 fc:01  1494   220  120  120        120         0        0               0             0              0               0    0       0      0           0 bash
    55a3c1068000 r--p 00114000 fc:01  1494    16   16   16         16        16        0               0             0              0               0    0       0      0           0 bash
    55a3c106c000 rw-p 00118000 fc:01  1494    36   36   36         36        36        0               0             0              0               0    0       0      0           0 bash
    55a3c1075000 rw-p 00000000 00:00     0    40   28   28         28        28        0               0             0              0               0    0       0      0           0
    55a3c18ba000 rw-p 00000000 00:00     0  1468 1400 1400       1400      1400        0               0             0              0               0    0       0      0           0 [heap]
    7f4488da7000 r--p 00000000 fc:01  3399    12   12    0         12         0        0               0             0              0               0    0       0      0           0 libnss_files-2.31.so
    7f4488daa000 r-xp 00003000 fc:01  3399    28   28    1         28         0        0               0             0              0               0    0       0      0           0 libnss_files-2.31.so
    7f4488db1000 r--p 0000a000 fc:01  3399     8    8    0          8         0        0               0             0              0               0    0       0      0           0 libnss_files-2.31.so
    7f4488db3000 r--p 0000b000 fc:01  3399     4    4    4          4         4        0               0             0              0               0    0       0      0           0 libnss_files-2.31.so
    7f4488db4000 rw-p 0000c000 fc:01  3399     4    4    4          4         4        0               0             0              0               0    0       0      0           0 libnss_files-2.31.so
    7f4488db5000 rw-p 00000000 00:00     0    24    0    0          0         0        0               0             0              0               0    0       0      0           0
    7f4488dc1000 r--p 00000000 fc:01  5822   200  120   15        120         0        0               0             0              0               0    0       0      0           0 LC_CTYPE
    7f4488df3000 r--p 00000000 fc:01  5827     4    4    0          4         0        0               0             0              0               0    0       0      0           0 LC_NUMERIC
    7f4488df4000 r--p 00000000 fc:01  5830     4    4    0          4         0        0               0             0              0               0    0       0      0           0 LC_TIME
    7f4488df5000 r--p 00000000 fc:01  5821  1404  140   25        140         0        0               0             0              0               0    0       0      0           0 LC_COLLATE
    7f4488f60000 r--p 00000000 fc:01  5825     4    4    0          4         0        0               0             0              0               0    0       0      0           0 LC_MONETARY
    7f4488f69000 r--s 00000000 fc:01  3656    28   28    2         28         0        0               0             0              0               0    0       0      0           0 gconv-modules.cache
    7f4488f70000 r--p 00000000 fc:01  5816  2968   64    7         64         0        0               0             0              0               0    0       0      0           0 locale-archive
    7f4489256000 rw-p 00000000 00:00     0    12    8    8          8         8        0               0             0              0               0    0       0      0           0
    7f4489259000 r--p 00000000 fc:01  3391   148  148    5        148         0        0               0             0              0               0    0       0      0           0 libc-2.31.so
    7f448927e000 r-xp 00025000 fc:01  3391  1504 1288   48       1288         0        0               0             0              0               0    0       0      0           0 libc-2.31.so
    7f44893f6000 r--p 0019d000 fc:01  3391   296  168    5        168         0        0               0             0              0               0    0       0      0           0 libc-2.31.so
    7f4489440000 ---p 001e7000 fc:01  3391     4    0    0          0         0        0               0             0              0               0    0       0      0           0 libc-2.31.so
    7f4489441000 r--p 001e7000 fc:01  3391    12   12   12         12        12        0               0             0              0               0    0       0      0           0 libc-2.31.so
    7f4489444000 rw-p 001ea000 fc:01  3391    12   12   12         12        12        0               0             0              0               0    0       0      0           0 libc-2.31.so
    7f4489447000 rw-p 00000000 00:00     0    16   16   16         16        16        0               0             0              0               0    0       0      0           0
```

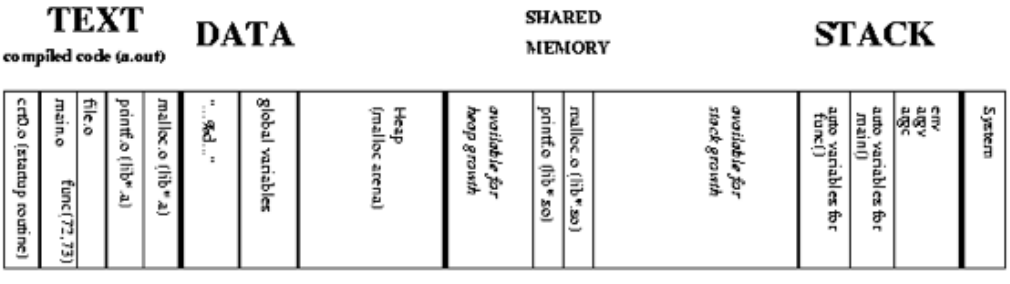
Second, I need to answer how many different entities make up a modern address space.

From information found in this source, we can write that modern linux memory space consists of the following:

- Text section
- Data section
- BSS Section
- A memory map of 0s
- An additional text, data and bss section for each shared library
- Any memory mapped files
- Any shared memory segments
- Any anonymous memmory mappings

<u>Notes</u>

- **Memory Layout of C programming**



8. Here I need to run `pmap` on program `question_4.c`, with different amounts of used memory.

   For it's solution, please referr to `question_8.c`.