

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 intuition that free memory should be less than total memory.

Notes

- I should ask professor the type of intuition 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 criteria 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 refer to `question_3.c`

Notes

- Learned that a large array (> 5000) can be generated using heap memory ^[1]
- **Command Line Arguments 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 arguments 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 refrerr 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.

```
ubuntu 1544 0.0 0.5 13896 5992 ? R 01:01 0:00 sshd: ubuntu@pts/0
ubuntu 1545 0.0 0.5 10032 5108 pts/0 Ss 01:01 0:00 -bash
ubuntu 1586 0.0 0.3 10600 3432 pts/0 R+ 01:13 0:00 ps aux
```

PID I will go for :)

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.

```

1545: ~bash
Address Perm Offset Device Inode Size Rss Pss Referenced Anonymous LazyFree SharedPaddedPapped Shared_MugetLib Private_MugetLib Swap SwapPss Locked ThPeligible Mapping
550c0f53000 r--p 00000000 fc:01 1494 100 100 100 100 0 0 0 0 0 0 0 0 0 0 0 0 0 bash
550c0f60000 r--p 00020000 fc:01 1494 700 700 700 700 0 0 0 0 0 0 0 0 0 0 0 0 bash
550c1031000 r--p 000a0000 fc:01 1494 220 120 120 120 0 0 0 0 0 0 0 0 0 0 0 0 bash
550c1060000 r--p 00110000 fc:01 1494 16 16 16 16 0 0 0 0 0 0 0 0 0 0 0 0 bash
550c1060000 r--p 00110000 fc:01 1494 36 36 36 36 0 0 0 0 0 0 0 0 0 0 0 0 bash
550c1070000 r--p 00000000 00:00 0 40 28 28 28 0 0 0 0 0 0 0 0 0 0 0 0 bash
550c1080000 r--p 00000000 00:00 0 1468 1400 1400 1400 1400 0 0 0 0 0 0 0 0 0 0 0 [heap]
7f4480007000 r--p 00000000 fc:01 3399 12 12 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 [libnss_files-2.31.so]
7f4480008000 r--p 00003000 fc:01 3399 28 28 1 28 0 0 0 0 0 0 0 0 0 0 0 0 0 [libnss_files-2.31.so]
7f448000b1000 r--p 00000000 fc:01 3399 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 [libnss_files-2.31.so]
7f448000b3000 r--p 0000c000 fc:01 3399 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 [libnss_files-2.31.so]
7f448000c1000 r--p 00000000 fc:01 3399 4 4 4 4 0 0 0 0 0 0 0 0 0 0 0 0 0 [libnss_files-2.31.so]
7f448000d2000 r--p 00000000 00:00 0 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 [libnss_files-2.31.so]
7f448000e1000 r--p 00000000 fc:01 5822 280 120 15 120 0 0 0 0 0 0 0 0 0 0 0 0 0 [LC_CTYPE]
7f448000f3000 r--p 00000000 fc:01 5827 4 4 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 [LC_NUMERIC]
7f448000f4000 r--p 00000000 fc:01 5830 4 4 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 [LC_TIME]
7f448000f5000 r--p 00000000 fc:01 5821 1404 140 25 140 0 0 0 0 0 0 0 0 0 0 0 0 0 [LC_COLLATE]
7f448000f6000 r--p 00000000 fc:01 5825 4 4 0 4 0 0 0 0 0 0 0 0 0 0 0 0 0 [LC_MONETARY]
7f448000f6000 r--p 00000000 fc:01 2656 20 20 2 20 0 0 0 0 0 0 0 0 0 0 0 0 0 [gnome-modules.cache]
7f448000f7000 r--p 00000000 fc:01 5816 2960 64 7 64 0 0 0 0 0 0 0 0 0 0 0 0 0 [locale-archive]
7f44800250000 r--p 00000000 00:00 0 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 [ ]
7f44800250000 r--p 00000000 fc:01 3391 140 140 5 140 0 0 0 0 0 0 0 0 0 0 0 0 0 [libc-2.31.so]
7f44800270000 r--p 00025000 fc:01 3391 1504 1200 40 1200 0 0 0 0 0 0 0 0 0 0 0 0 0 [libc-2.31.so]
7f448003f6000 r--p 00190000 fc:01 3391 296 160 5 160 0 0 0 0 0 0 0 0 0 0 0 0 0 [libc-2.31.so]
7f44809400000 ---p 001e7000 fc:01 3391 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 [libc-2.31.so]
7f44809440000 r--p 001e7000 fc:01 3391 12 12 12 12 12 12 0 0 0 0 0 0 0 0 0 0 0 [libc-2.31.so]
7f44809440000 r--p 001e0000 fc:01 3391 12 12 12 12 12 12 0 0 0 0 0 0 0 0 0 0 0 [libc-2.31.so]
7f44809447000 r--p 00000000 00:00 0 16 16 16 16 16 16 0 0 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

Notes

- Memory Layout of C programming

TEXT compiled code (.a.out)	DATA	SHARED MEMORY	STACK
<div> <div> crt0.o (startup routine)</div> <div>main.o func(72, 73)</div> <div>file.o func(72, 73)</div> <div>printf.o (libc.a)</div> <div>malloc.o (libc.a)</div> <div>.._Sd...</div> <div>global variables</div> </div> <div> <div>heap (malloc arena)</div> </div>	<div> <div>available for heap growth</div> </div>	<div> <div>malloc.o (libc.so)</div> <div>printf.o (libc.so)</div> </div>	<div> <div>available for stack growth</div> <div>auto variables for main()</div> <div>auto variables for func()</div> <div>env argv argc</div> <div>System</div> </div>

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

For it's solution, please refrerr to question_8.c.