CS4375-13948 Fall 2023
Omar Hoang
oahoang@miners.utep.edu

Homework Report
Submitted on 11/10/23

HW 4: Lazy Allocation for xv6

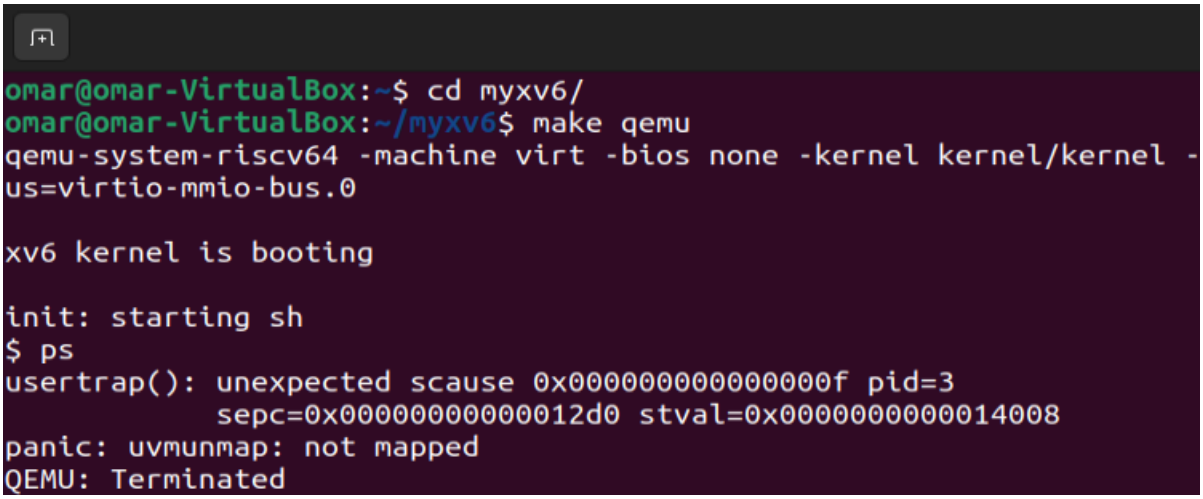
Task 1. freepmem() system call

```
xv6 kernel is booting

init: starting sh
$ free
133390336
$ free -k
130264
$ free -m
127
$ memory-user 1 4 1 &
$ allocating 0x0000000000000001 mebibytes
malloc returned 0x0000000000003010
freeing 0x0000000000000001 mebibytes
allocating 0x0000000000000002 mebibytes
malloc returned 0x0000000000103020
freeing 0x0000000000000002 mebibytes
allocating 0x0000000000000003 mebibytes
malloc returned 0x0000000000003020
freeing 0x0000000000000003 mebibytes
allocating 0x0000000000000004 mebibytes
malloc returned 0x0000000000303030
freeing 0x0000000000000004 mebibytes
```

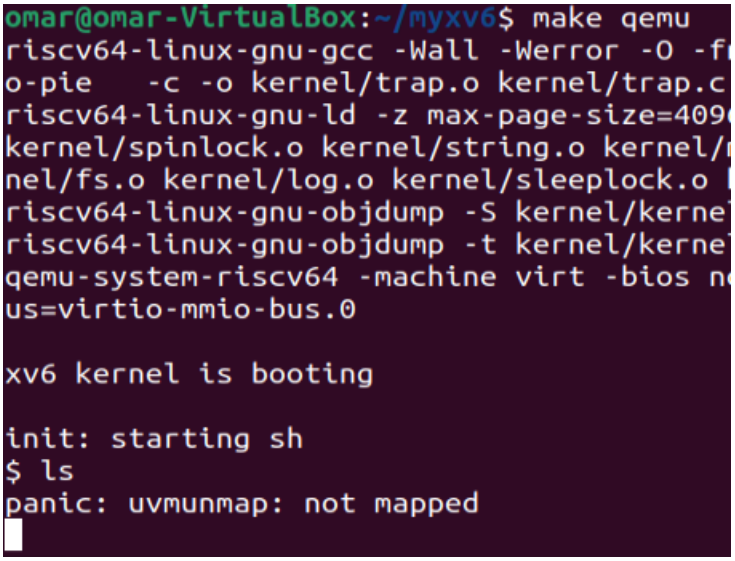
After implementing freepmem() into myxv6 I was able to see how much free memory I have. I was able to see the results by running free.c & memory-user.c. Where I could see the free memory being allocated when giving values while using the commands “free” and “memory -user”

Task 2. Change sbrk() so that it does not allocate physical memory.



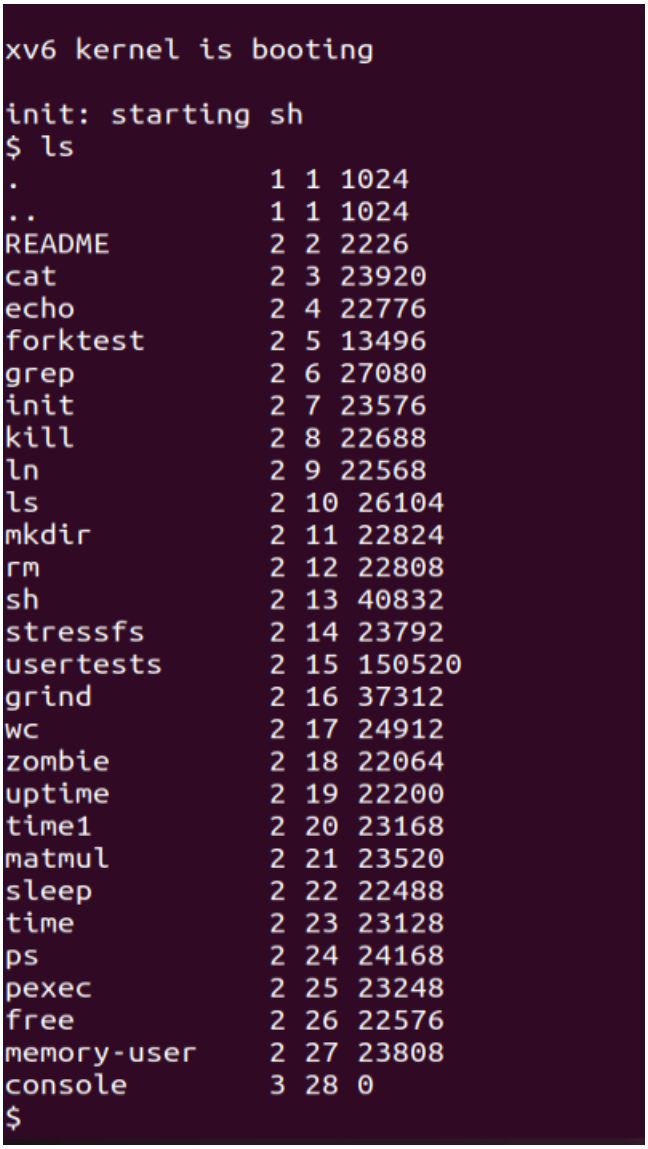
After modifying sys_sbrk() I was given two errors. One in the usertrap() and another with uvmunmap().

Task 3. Handle the load and store faults that result from Task 2



I fixed the error in usertrap() by handling the fault codes that were causing the error. Those being the load and store faults. After fixing the faults I was still getting another error with the uvmunmap() call.

Task 4. Fix kernel panic and any other errors.



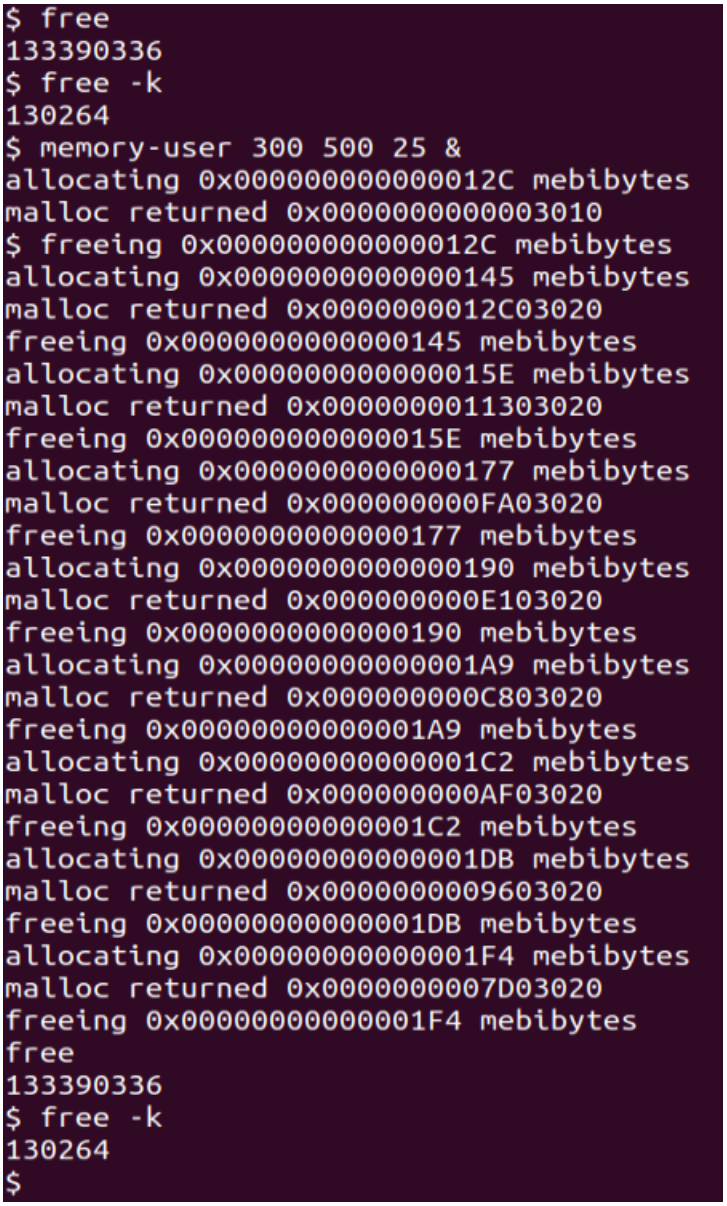
I was able to fix the uvmunmap() by going to where that panic call would happen in vm.c and just let it continue rather than panic, but myxv6 was still not running commands. In order to fix these errors, instead of just correcting uvmunmap() panic error I had to do the same with uvmcopy(). Then I was able to do commands.

Task 5. Test your lazy memory allocation.

```
xv6 kernel is booting

init: starting sh
$ memory-user 1 24 6 &
$ allocating 0x0000000000000001 mebibytes
malloc returned 0x0000000000003010
freeing 0x0000000000000001 mebibytes
allocating 0x0000000000000007 mebibytes
malloc returned 0x0000000000103020
freeing 0x0000000000000007 mebibytes
allocating 0x000000000000000D mebibytes
malloc returned 0x0000000000803030
freeing 0x000000000000000D mebibytes
allocating 0x0000000000000013 mebibytes
malloc returned 0x0000000000203030
freeing 0x0000000000000013 mebibytes
```

```
$ free -k
130264
$ memory-user 64 100 10 &
allocating 0x0000000000000040 mebibytes
malloc returned 0x0000000000003010
$ freeing 0x0000000000000040 mebibytes
allocating 0x000000000000004A mebibytes
malloc returned 0x0000000004003020
freeing 0x000000000000004A mebibytes
allocating 0x0000000000000054 mebibytes
malloc returned 0x0000000003603020
freeing 0x0000000000000054 mebibytes
freeallocating 0x000000000000005E mebibytes
malloc returned 0x0000000002C03020
freeing 0x000000000000005E mebibytes
```



Here is how I tested the lazy allocation with different values using memory-user command.