CS4375-13948 Fall 2023 Homework Report

**Edmundo Pariente** Submitted on **11/10/2023**

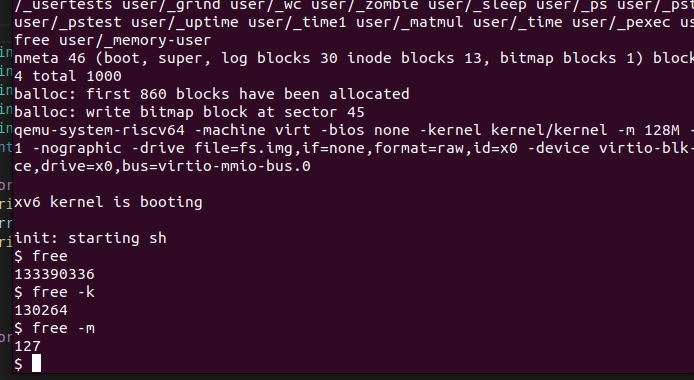
**eipariente@miners.utep.edu**

**HW 4: Lazy Allocation for xv6**

Task 1. freepmem() system call

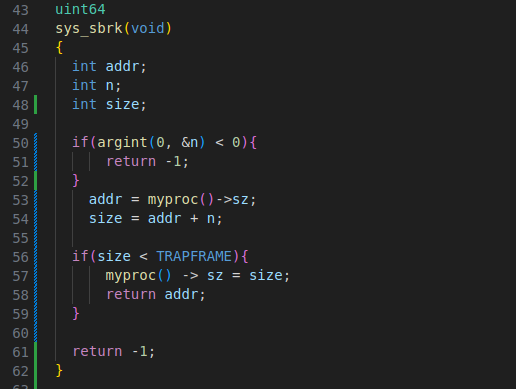
For this task I created a new system call, I needed to change add the specific function definitions inside sysproc.c, usys.pl, user.h syscall.h and kalloc.c. Finally adding the system call to the Makefile so that the operating system would be able to recognize it.

The process was similar to the system calls that I previously had to implement in the other assignments so it wasn’t much trouble, it helped to reinforce the knowledge of how these types of calls are implemented.



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

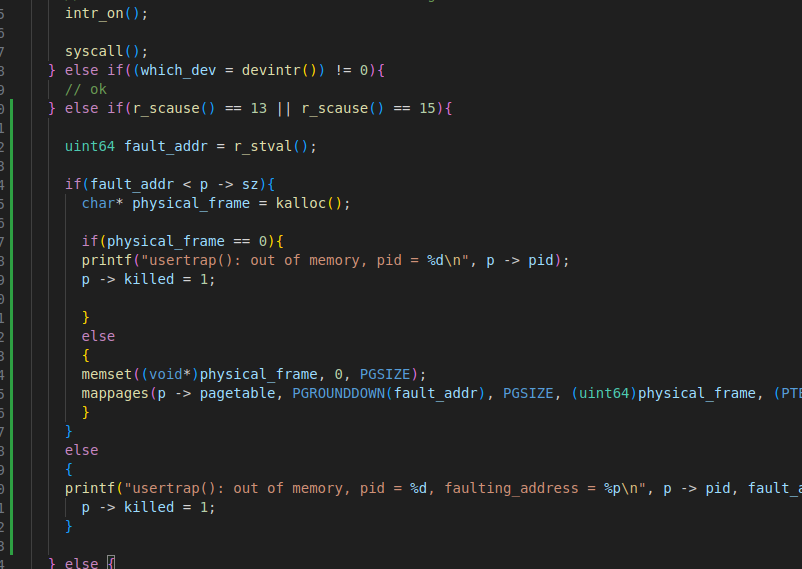
I was getting some errors about incorrect mapping of the memory tables, I think I corrected this by changing the code inside the usertrap method in trap.c.



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

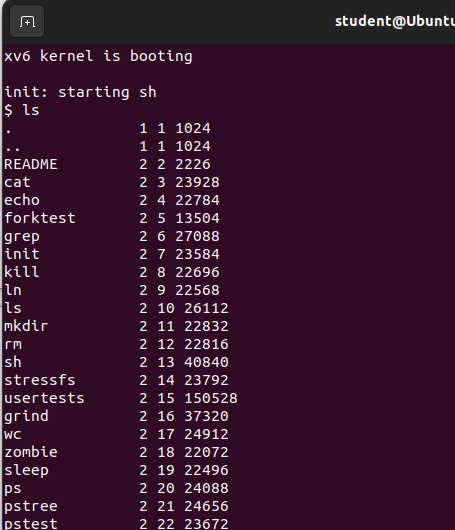
As mentioned in the previous task 2, the errors I was getting were a result of incorrect mapping, the operating system was preventing from correctly executing qemu in the command line and was unable to do anything else.

By modifying the usertrap function inside trap.c I was able to fix it and correct the behavior of this process.

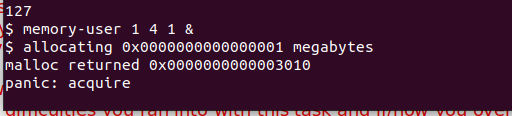


Task 4. Fix kernel panic and any other errors.

The problem here was mostly the panic errors that I was getting but in this case I don’t think they are relevant because we are dealing with lazy allocation, to get rid of them I simply had to change some functions in the vm.c file that deal with the uvmunmap and uvmcopy methods to map and copy the addresses that will be manipulated for the desired process.



Task 5. Test your lazy memory allocation.



I’m just showing here the command that I implemented of memory user call to show that the changes made throughout the entire process were correct and displaying the right information.

After executing the command it will tell you the amount of megabytes allocated and the malloc return value.