CS4375-13948 Fall 2023 Homework Report
David Vasquez
11/10/2023
<Your email address>

HW 4: Lazy Allocation for xv6

Please replace red text with your report text and any tables or figures, names of any accompanying files, etc. Remember to commit all the files for your lab submission, to put the URL for your private xv6 repo in the Teams assignment, to submit the Teams assignment, and to give the instructor and TA access to your repo.

Task 1. freepmem() system call

Your modified files to implement the freepmem() system call should be in the hw4 branch of your xv6 repo. Show and explain the results of testing the freepmem() system call using the provided free command and memory-user program. Summarize what you learned by carrying out this task.

Freepmem() calculates and returns the amount of free physical memort in bytes , this is done by traversing through a linked list of free memory blocks then multiplying yhe number of free pages by the page size. When free calls freepmem() it gives the user options to either to receive the amount of memory in kilobytes or megabytes. In doing this task I learned about accessing the freepages and calculating the amount of free space using these freepages from traversing linked list of free memory blocks using pointers.

```
freefreeing 0x0000000000000003 mebibytes

130199552
$ allocating 0x0000000000000004 mebibytes
malloc returned 0x0000000000303030
freeing 0x0000000000000004 mebibytes
free
133390336
$ memory-user 1 8 1 &
$ allocating 0x0000000000000001 mebibytes
malloc returned 0x0000000000003010
free
132300800
$ freeing 0x0000000000000001 mebibytes
allocating 0x0000000000000002 mebibytes
malloc returned 0x0000000000103020
free
130199552
$ freeing 0x0000000000000002 mebibytes
allocating 0x0000000000000003 mebibytes
malloc returned 0x0000000000003020
free
130199552
$ freeing 0x0000000000000003 mebibytes
allocating 0x0000000000000004 mebibytes
malloc returned 0x0000000000303030
free
125997056
$ freeing 0x0000000000000004 mebibytes
allocating 0x0000000000000005 mebibytes
malloc returned 0x0000000000203030
freeing 0x0000000000000005 mebibytes
allocating 0x0000000000000006 mebibytes
malloc returned 0x0000000000103030
freeing 0x0000000000000006 mebibytes
allocating 0x0000000000000007 mebibytes
malloc returned 0x0000000000003030
free
125997056
$ freeing 0x0000000000000007 mebibytes
allocating 0x0000000000000008 mebibytes
malloc returned 0x0000000000703040
freeing 0x0000000000000008 mebibytes
free
133390336
$
```

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

Commit and push your modified sysproc.c file containing your changed sbrk() system call implementation to your hw4 branch. Show the results of starting xv6 and attempting to run a user command. Explain why the errors occur. Summarize what you learned by carrying out this task.

The error message was caused because there was an issue when the kernel tried to unmap a region that was not previously mapped yet. I learned how to create virtual memory on the disk by editing the sbrk().

```
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-devi
ce,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ free
usertrap(): unexpected scause 0x000000000000000f pid=3
            sepc=0x0000000000012b8 stval=0x0000000000004008
panic: uvmunmap: not mapped
QEMU: Terminated
```

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

Commit and push your modified trap.c file containing your modified usertrap() function to your hw4 branch. Show the results of starting xv6 and running a user command. Explain why errors occur.

Summarize what you learned by carrying out this task.

Describe any difficulties you ran into with this task and if/how you overcame them.

```
david321239@ubuntu1:~/Desktop/Os$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-devi
ce,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ free
panic: uvmunmap: not mapped
```

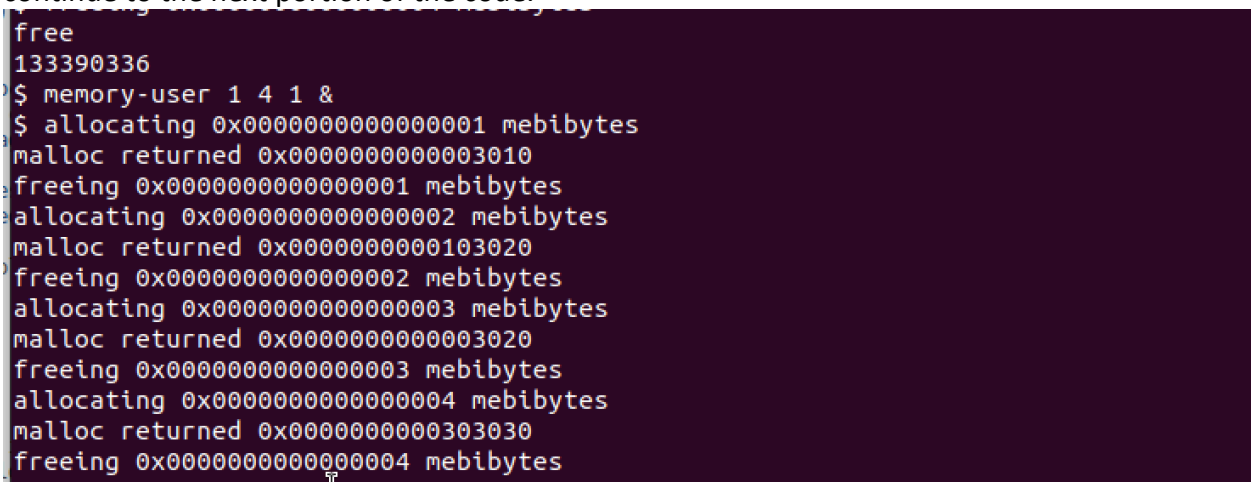
Task 4. Fix kernel panic and any other errors.

List what files you changed to fix the errors and describe the changes. Commit and you're your modifications to your hw4 branch. Show the results of starting xv6 and running user commands after you have fixed the errors you found.

Summarize what you learned by carrying out this task.

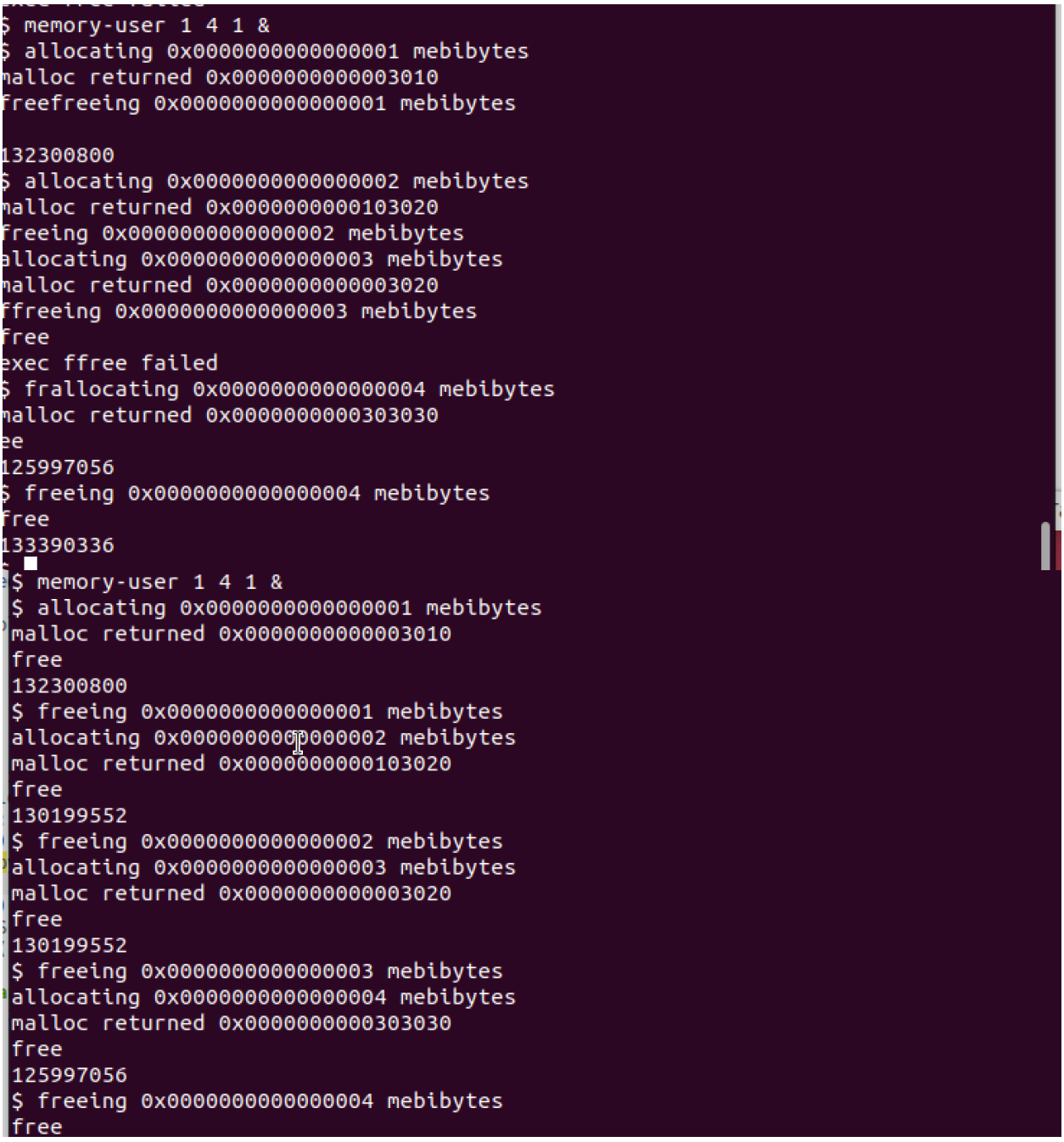
Describe any difficulties you ran into with this task and if/how you overcame them.

To fix the errors we had to replace some code in the vm.c kernel code. The first instance being changing the behavior when the page table entry is marked as valid, instead of panicking when the virtual address is not mapped, it will skip the panicking and continuing to the next portion of the code. After that we change another piece of code in vm.c that is futher down. We change the portion where it check if there is a page is present. Originally the code would check if a page is present, and if it not, it panics. We change this so that if there is no page, then we just continue to the next portion of the code.



Task 5. Test your lazy memory allocation.

Describe the purpose of each of your test cases and show the results. Commit and push your test programs to the user directory in your hw4 branch.


$ memory-user 1 4 1 &
$ allocating 0x0000000000000001 mebibytes
malloc returned 0x0000000000003010
freefreeing 0x0000000000000001 mebibytes
132300800
$ allocating 0x0000000000000002 mebibytes
malloc returned 0x0000000000103020
freeing 0x0000000000000002 mebibytes
allocating 0x0000000000000003 mebibytes
malloc returned 0x0000000000003020
ffreeing 0x0000000000000003 mebibytes
free
exec ffree failed
$ frallocating 0x0000000000000004 mebibytes
malloc returned 0x0000000000303030
ee
125997056
$ freeing 0x0000000000000004 mebibytes
free
133390336
$ memory-user 1 4 1 &
$ allocating 0x0000000000000001 mebibytes
malloc returned 0x0000000000003010
free
132300800
$ freeing 0x0000000000000001 mebibytes
allocating 0x0000000000000002 mebibytes
malloc returned 0x0000000000103020
free
130199552
$ freeing 0x0000000000000002 mebibytes
allocating 0x0000000000000003 mebibytes
malloc returned 0x0000000000003020
free
130199552
$ freeing 0x0000000000000003 mebibytes
allocating 0x0000000000000004 mebibytes
malloc returned 0x0000000000303030
free
125997056
$ freeing 0x0000000000000004 mebibytes
free

```
$ 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
```

Extra Credit Task 6. Enable use of the entire virtual address space.
List what files you changed and describe the changes.
Summarize what you learned by carrying out this task.

Extra Credit Task 7. Allow a process to turn memory overcommitment on and off.
Describe your approach to this task. Commit and push your code changes to your hw4 branch.
Show the results of a test program that turns memory overcommitment on and off.
Summarize what you learned by carrying out this task.