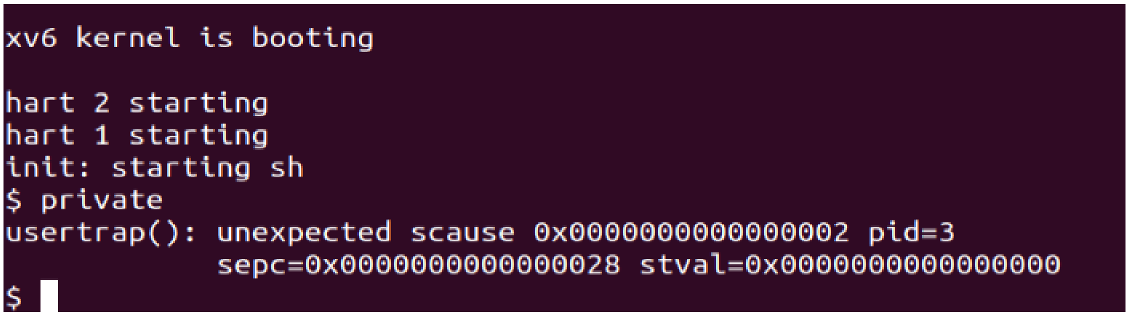
CS4375-13948 Fall 2023
Omar Hoang
oahoang@miners.utep.edu

Homework Report
Submitted on 12/8/2023

UG HW5: Anonymous Memory Mappings for xv6

Task 1.

a.



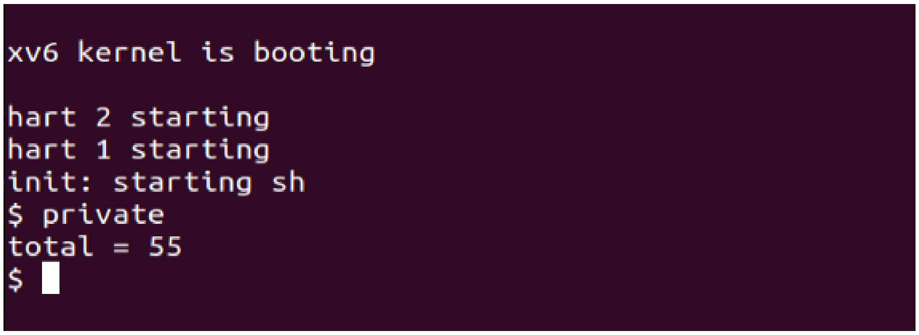
```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ private
usertrap(): unexpected scause 0x0000000000000002 pid=3
            sepc=0x0000000000000028 stval=0x0000000000000000
$ 
```

mmap() is creating a shared memory region and munmap() is releasing the memory after private.c has been done executing.

The reason the program aborts is because of a store page fault that being because of permissions that are not being checked.

b.

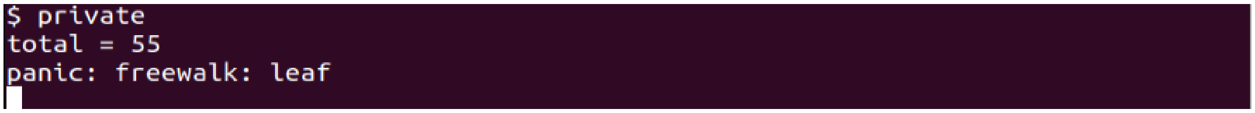


```
xv6 kernel is booting

hart 2 starting
hart 1 starting
init: starting sh
$ private
total = 55
$ 
```

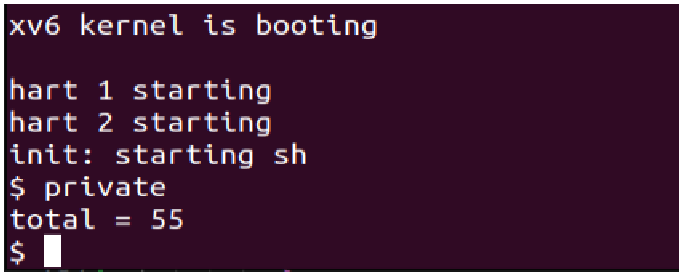
I fixed the usertrap() fault by going through the mapped pages with a for-loop and making sure each mapped page had the correct permissions. The only difficulty I had was adding the checks for the permissions. I overcame this by just including those checks in a if-statement within the for-loop.

c.



```
$ private
total = 55
panic: freewalk: leaf
```

When commenting out munmap() the reason the panic is outputted is because the memory is not being unmapped after the program is done running.



```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ private
total = 55
$ 
```

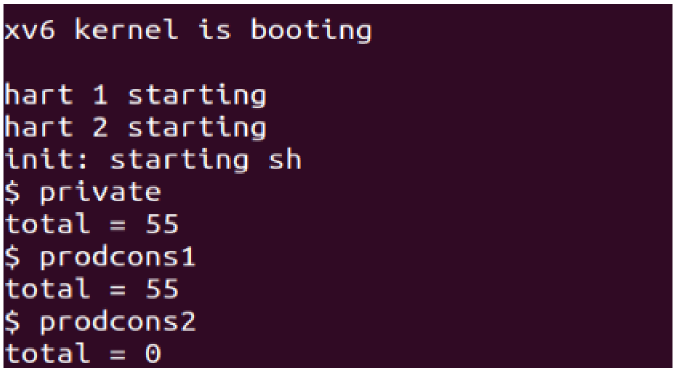
After adding the code the panic was lifted. The conditions needed for the physical memory for a mapped memory region to be freed is the mapped memory exists, the memory can be shared, there are private memory regions, and cleaning up the memory once a process is completed.

Task 2.

a. uvmcopy() copies the virtual memory from the parent process to the child process when the memory is not going to be shared. Uvmcopyshared() copies the virtual memory from the parent process to the child process but only if the mapped memory is being shared.

b. If any of the flags from the memory mapped region comes back as PRIVATE then uvmcopy() will be called to copy the virtual memory among the process, and if the flags come back as SHARED then uvmcopyshared() will be called to copy the virtual memory in the shared regions.

c.



```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ private
total = 55
$ prodcons1
total = 55
$ prodcons2
total = 0
```

They produce different results because prodcons1 is mapping memory regions that are shared so the process will be shared and seeing through the producer and consumer. Prodcons2 memory regions are set as private so the producer and consumer will not see each other's so the total will output as what it was initialized.

Task 3.

a. It produces the wrong results because the parent process will not see the child process changes when fork() is done. The implementation as of right now only handles file spaces not the parents.

b.

Summary:
I have learned how the OS (xv6) is able to create the illusion of a bigger virtual space through memory mapping techniques. It is a very tedious task where you need to able to trace where you are in the memory region in order to get the correct results.