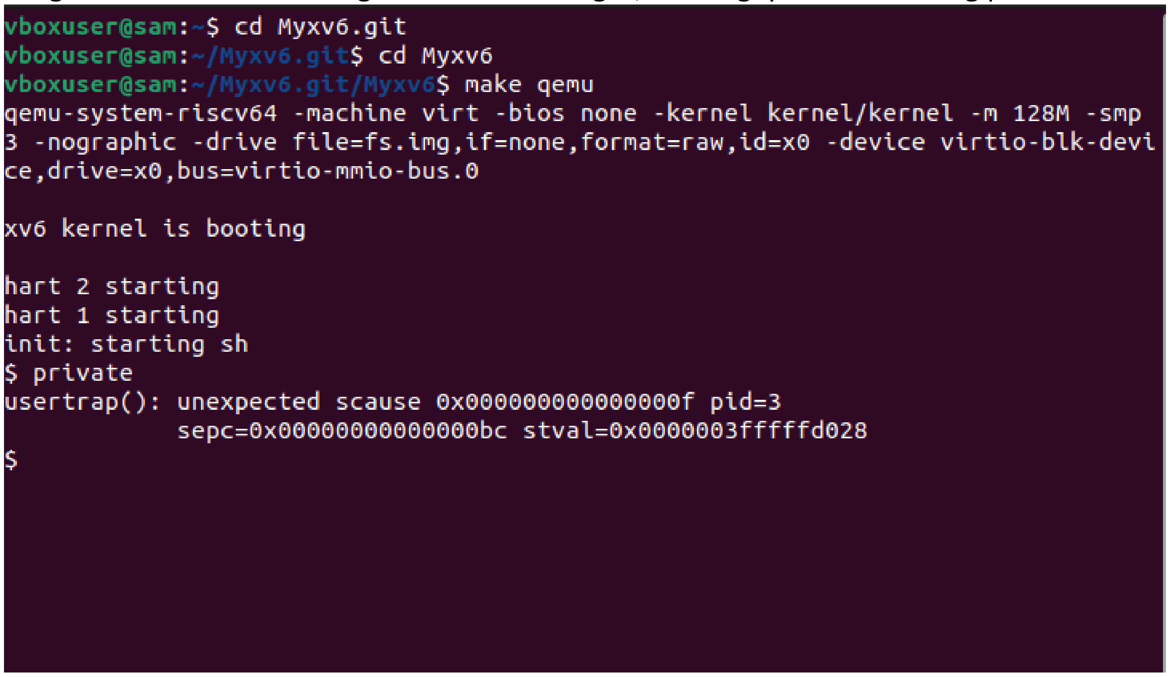
Fall 2023 Report
Samuel Ramírez 12/1/2023

Anonymous Memory Mappings for xv6

Task 1.

a. Show the results of running the private program.

Program aborts after adding the task1.a changes, running qemu and calling private:



```
vboxuser@sam:~$ cd Myxv6.git
vboxuser@sam:~/Myxv6.git$ cd Myxv6
vboxuser@sam:~/Myxv6.git/Myxv6$ 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 2 starting
hart 1 starting
init: starting sh
$ private
usertrap(): unexpected scause 0x000000000000000f pid=3
            sepc=0x00000000000000bc stval=0x0000003fffffd028
$
```

Figure 1: Calling private after task1.a changes.

Describe how the private program uses mmap() and munmap().

Private.c is using mmap() in order to create a shared memory space in which parts of the program can exchange and share information.

The munmap() function is being used to release the shared memory that was created with mmap() in order to tell the system that the program is done using that portion of memory and can be used for new different tasks.

They are both being using to allocate share memory for the buffer structure and lets the producer and consumer parts of the program to interact and munmap() is releasing the allocated memory when the program is done using it.

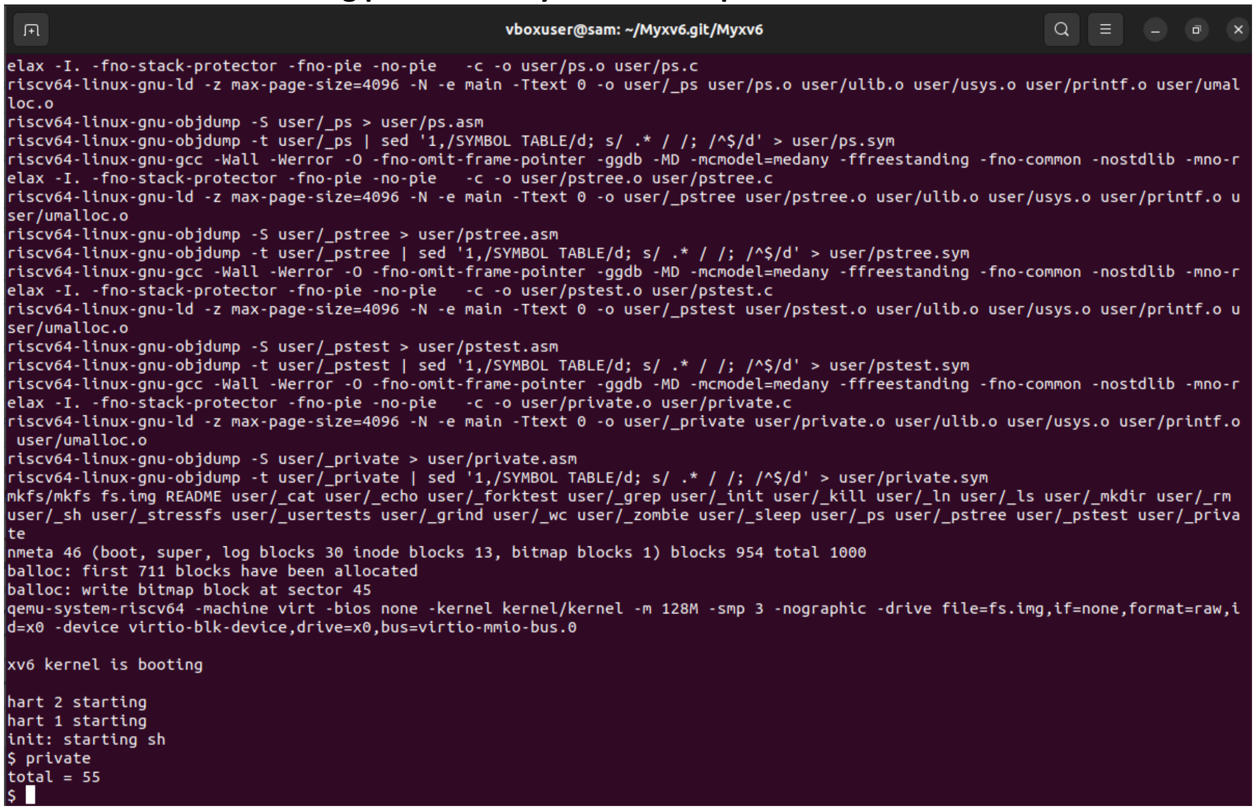
Explain why the program aborted before you modified usertrap() in kernel/trap.c for part b.

The user trap error is about a fault that is occurring in the user space. The program is being affected by a segmentation fault which is being load or store faults which are not being handle properly.

b. Describe how you fixed the usertrap() from part a.

In order to fix the user trap error from task1a I had to modify the usertrap() function in order for it to handle page faults in the mapped memory region. Whenever a load or store fault occurs it will check if its inside a mapped memory region and verifies all the necessary permissions and then allocates physical memory frame and finally maps it into the process's page table.

Show the result of running private after you fixed the problem.



vboxuser@sam: ~/Myxv6.git/Myxv6

```
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/ps.o user/ps.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_ps user/ps.o user/ulib.o user/usys.o user/printf.o user/umal
loc.o
riscv64-linux-gnu-objdump -S user/_ps > user/ps.asm
riscv64-linux-gnu-objdump -t user/_ps | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/ps.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-r
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/pstree.o user/pstree.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_pstree user/pstree.o user/ulib.o user/usys.o user/printf.o u
ser/umalloc.o
riscv64-linux-gnu-objdump -S user/_pstree > user/pstree.asm
riscv64-linux-gnu-objdump -t user/_pstree | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/pstree.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-r
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/pstest.o user/pstest.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_pstest user/pstest.o user/ulib.o user/usys.o user/printf.o u
ser/umalloc.o
riscv64-linux-gnu-objdump -S user/_pstest > user/pstest.asm
riscv64-linux-gnu-objdump -t user/_pstest | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/pstest.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-r
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/private.o user/private.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_private user/private.o user/ulib.o user/usys.o user/printf.o
 user/umalloc.o
riscv64-linux-gnu-objdump -S user/_private > user/private.asm
riscv64-linux-gnu-objdump -t user/_private | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/private.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln user/_ls user/_mkdir user/_rm
user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_sleep user/_ps user/_pstree user/_pstest user/_priva
te
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
balloc: first 711 blocks have been allocated
balloc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,i
d=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

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

Figure 2: Running private after fixing the user trap error.

Describe any difficulties you encountered with this task and how you overcame them.
how is this: When doing this task and following the homework pdf didn't work for me at first. I had trouble with some parts of those instructions, for example: When using PGROUNDDOWN I wasn't sure how to round down the fault address and map a new frame into the process's page table. To overcome this, I had to search and learn how PGROUNDDOWN worked and how I was supposed to use it. At the end I felt like there were to many changes that had to be made to fix this user trap error that I had a hard time debugging the new errors that appeared or didn't do anything.

c. **Explain why running the private program with the call to munmap() commented out initially caused a kernel panic before you added the code for part c to freeproc().**

When running the private.c program, and after commenting out the munmap() call, it causes a kernel panic because that change is preventing the release of resources that are connected to the mapped memory region. The kernel expects a proper termination of the processes with allocated resources being properly released.

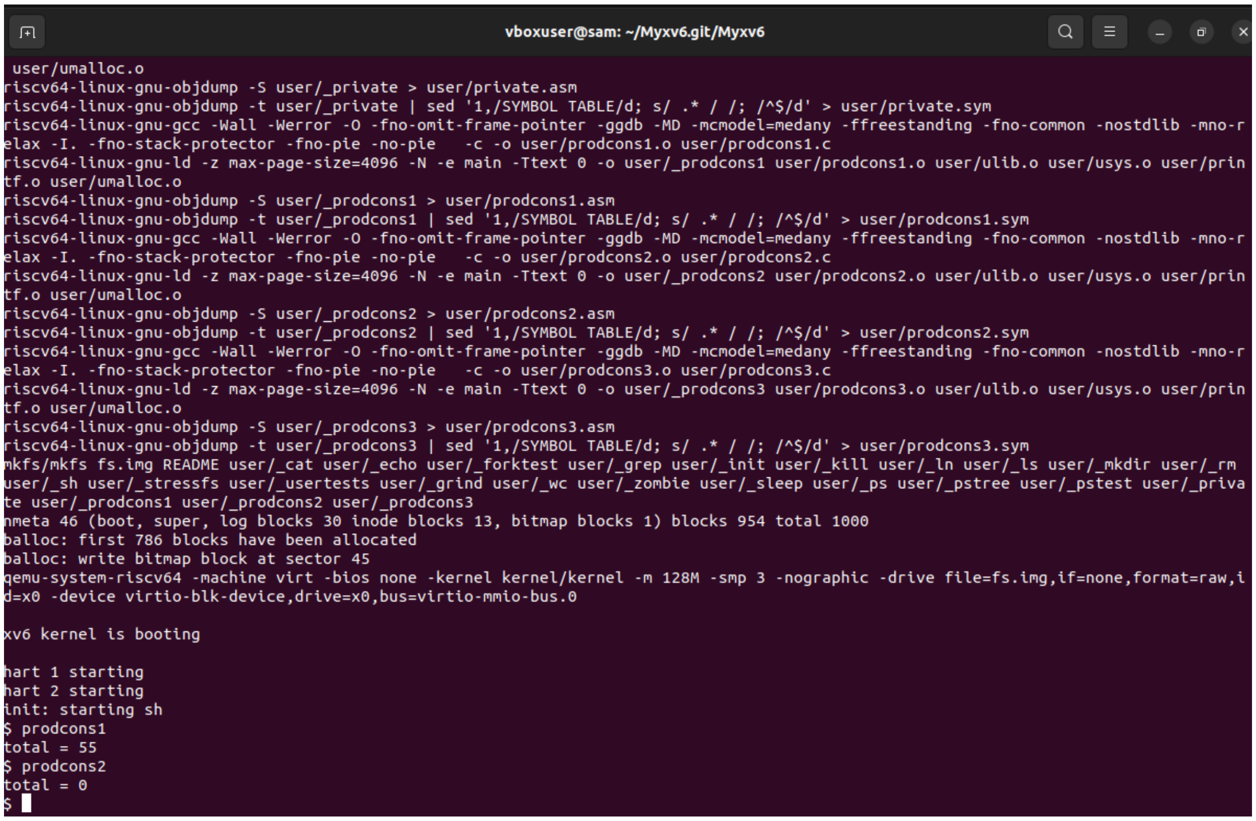
Under what conditions does the physical memory for a mapped memory region need to be freed in freeproc() and what happens if this isn't done?
The physical memory for a mapped memory region needs to be freed under the condition of private mapping when it's created with a MAP_PRIVATE flag and when it has shared mapping with the MAP_SHARED flag. If the conditions are not met, it might mean that the mapped memory region is being used by other processes, which means that the physical memory associated should not be freed.

Task 2.

a. **Explain the difference between uvmcopy() and uvmcopyshared().**
The differences between uvmcopy() and uvmcopyshared() is that the uvmcopy function is creating a new copy of the memory content for a new process, while uvmcopyshared is sharing the same physical memory content between old and the new address spaces.

b. **Explain how the new code added to fork() works for PRIVATE and SHARED mapped regions, respectively.**
The new added code in fork() is making sure that the child process has the same mapped memory's regions as the parent. They are both handling PRIVATE and SHARED mappings, for shared mapping it establishes a relationship between the parent and child process for the shared regions.

c. **Show the results of running the prodcons1 and prodcons2 programs. Explain the results, including why they produce different results.**



```
vboxuser@sam: ~/Myxv6.git/Myxv6
 user/umalloc.o
riscv64-linux-gnu-objdump -S user/_private > user/private.asm
riscv64-linux-gnu-objdump -t user/_private | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/private.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-r
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/prodcons1.o user/prodcons1.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_prodcons1 user/prodcons1.o user/ulib.o user/usys.o user/prin
tf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_prodcons1 > user/prodcons1.asm
riscv64-linux-gnu-objdump -t user/_prodcons1 | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/prodcons1.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-r
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/prodcons2.o user/prodcons2.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_prodcons2 user/prodcons2.o user/ulib.o user/usys.o user/prin
tf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_prodcons2 > user/prodcons2.asm
riscv64-linux-gnu-objdump -t user/_prodcons2 | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/prodcons2.sym
riscv64-linux-gnu-gcc -Wall -Werror -O -fno-omit-frame-pointer -ggdb -MD -mcmodel=medany -ffreestanding -fno-common -nostdlib -mno-r
elax -I. -fno-stack-protector -fno-pie -no-pie   -c -o user/prodcons3.o user/prodcons3.c
riscv64-linux-gnu-ld -z max-page-size=4096 -N -e main -Ttext 0 -o user/_prodcons3 user/prodcons3.o user/ulib.o user/usys.o user/prin
tf.o user/umalloc.o
riscv64-linux-gnu-objdump -S user/_prodcons3 > user/prodcons3.asm
riscv64-linux-gnu-objdump -t user/_prodcons3 | sed '1,/SYMBOL TABLE/d; s/ .* / /; /^$/d' > user/prodcons3.sym
mkfs/mkfs fs.img README user/_cat user/_echo user/_forktest user/_grep user/_init user/_kill user/_ln user/_ls user/_mkdir user/_rm
user/_sh user/_stressfs user/_usertests user/_grind user/_wc user/_zombie user/_sleep user/_ps user/_pstree user/_pstest user/_priva
te user/_prodcons1 user/_prodcons2 user/_prodcons3
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 954 total 1000
balloc: first 786 blocks have been allocated
balloc: write bitmap block at sector 45
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,i
d=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

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

Figure 3: Running prodcons 1 and 2 after task2.a – b changes.

They produce different results because prodcons1 is using MAP_SHARED, so multiple processes share the same memory region, and prodcons2 is using MAP_PRIVATE, where each process gets its own private copy of the memory region.
The result for prodcons1 is 55 because it's the total sum of all the produced items being consumed by the consumer process. Result for prodcons2 is 0 because the processes have their own private memory and are not sharing any produced items between the producer and consumer processes.

Task 3.

a. **Explain why the prodcons3 program produces incorrect results before you implement part b**.
 It seems that prodcons3 has an issue where the producer and consumer might not be synchronizing which affects the termination of processes. The producer is not updating the new index for its buffer. I would have to implement a way to handle this situation on task3.b

b. **(Extra credit) Show the result of running prodcons3 after you have completed Task 3b. Describe any difficulties you ran into with this task and how you overcame them.**

Summary:
From working on this homework assignment, I was able to learn more about shared memory management and how processes synchronization works. In task 1 I learned how to fix a

segmentation fault and how to debug other problems and errors. The second step I learned the how uvmmcopy and uvmcopyshared worked and their differences. In the last task I learned why the implementation of prodcons3 didn’t work and why it’s important for the producer and consumer to be synchronized.