CS4375-13948 Fall 2023 Homework Report
David Vasquez
Submitted on 12/8/2023
Davasquez7@miners.utep.com

UG HW5: Anonymous Memory Mappings for xv6

Please replace red text with your report text. Remember to commit all the files for your 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.

a. The private program uses the mmap() to create a shared memory region, then munmap() is then used to release that memory region after the program has finished using it. The error occurred because the kernel cannot allocate physical memory for the requested virtual memory mapping. This is because in this case the usertrap only checks weather the faulting address is within the valid range of the virtual space.

```
xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$ private
usertrap(): invalid memory access, pid=3, faulting_address=0x0000003fffffd028
$
```

b. The error is solved by editing the usertrap by adding additional checks to beyond just checking the range. The new code loops the mapped regions(p->mmr) and check if the faulting address is with the valid range of any specific region. It also checks the protection settings of the mapped region to make sure the requested operation is allowed (PROT_READ).

```
xv6 kernel is booting

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

c. When commenting out munmap() the kernel panics because the program expects handling deallocation of resources associated with the process, munmap() is crucial for releasing memory regions mapped to the process. There are three conditions that must be met, MAP_PRIVATE flag is set, the mapped region has no other family members, and when the mapped region is apart of a family of mappings in which the process is removed from the family structure, and the physical memory is deallocated only when

the last process in the family releases its reference. Faliure to free physical memory can lead in memory leaks and increased resource usage, that leads to performance degration.

Task 2.

a. Uvmcopy() allocates new physical pages for each copied page creating private copies, it copies the content from the original physical page to the new page and establishes a new mapping. The uvmcopyshared() directly maps the original physical pages from the source to the destination table creating the shared mappings.
b. In prodcon1 the functions are called in order in the same process. Both of the functions run in a single threaded manner and the comsumer goes after the producer completes. Compared to the prodcon2, in which two processes are created using the fork function, one for the producer and one for the consumer. These processes run concurently.

```
david321239@ubuntu1:~/Desktop/Os2$ 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-de
vice,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

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

Task 3.

a.

```
$ prodcons3
total = 673720320
$
```

Prodcons 3 most likely does not work most likely to its lack of synchronization and a the absence of a waiting mechanism. The program manipulates shared variables with out any synchronization mechanisms. This lead to multiple processes accessing and modifying these variables at the same time (race condition). There is no mechanism to stop the consumer from consuming before the producer has produced or the other way around. This could probably solved with implementing semaphores.

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:

By completing homework 5, I gained a better understanding on how memory mapping works and how to implement it into our program. This was done by applying the mmap() and the

munmap() systemcalls. As well I recognized what is needed to carry out these systems calls how different conditions affects how theses programs run.